Arquitectura de Computadores (AC)

2º curso / 2º cuatr. Grado Ing. Inform.

Cuaderno de prácticas. Bloque Práctico 4. Optimización de código

Estudiante (nombre y apellidos): Clara María Romero Lara Grupo de prácticas y profesor de prácticas: D1, Niceto Luque

Fecha de entrega: 12-05-2020

Fecha evaluación en clase: 22-05-2020

Antes de comenzar a realizar el trabajo de este cuaderno consultar el fichero con los normas de prácticas que se encuentra en SWAD

Denominación de marca del chip de procesamiento o procesador (se encuentra en /proc/cpuinfo): *Intel(R) Core(TM) i5-8250U CPU @ 1.60GHz*

Sistema operativo utilizado: Arch Linux 5.5.13-arch2-1 Versión de gcc utilizada: gcc (Arch Linux 9.3.0-1) 9.3.0

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

```
→ lscpu
Architecture:
                                                                       32-bit, 64-bit
Little Endian
CPU op-mode(s):
Byte Order:
                                                                       39 bits physical, 48 bits virtual
Address sizes:
CPU(s):
On-line CPU(s) list:
Thread(s) per core:
Core(s) per socket:
Socket(s):
NUMA node(s):
Vendor ID:
CPU family:
Model:
Model name:
                                                                       GenuineIntel
                                                                       Intel(R) Core(TM) i5-8250U CPU @ 1.60GHz
modet name:
Stepping:
CPU MHz:
CPU max MHz:
CPU min MHz:
BogoMIPS:
Virtualization:
                                                                       10
2635.982
                                                                       3400.0000
400.0000
                                                                       3601.00
VT-x
                                                                       128 KiB
128 KiB
Lld cache:
Lli cache:
                                                                       1 MiB
6 MiB
0-7
L2 cache:
L3 cache:
L3 cache:
NUMA node0 CPU(s):
Vulnerability Itlb multihit:
Vulnerability L1tf:
Vulnerability Mds:
Vulnerability Meltdown:
Vulnerability Spec store bypass:
Vulnerability Spectre v1:
Vulnerability Spectre v2:
Vulnerability Tsx async abort:
Flans:
                                                                       0-7
KVM: Mitigation: Split huge pages
Mitigation: PTE Inversion; VMX conditional cache flushes, SMT vulnerable
Vulnerable: Clear CPU buffers attempted, no microcode; SMT vulnerable
Mitigation; PTI
Mitigation: Speculative Store Bypass disabled via protl and seccomp
                                                                                                Speculative Store Bypass disabled via prctl and seccomp
                                                                       Mitigation;
                                                                       Mitigation;
Mitigation;
                                                                                               usercopy/swapgs barriers and <u>user pointer sanitization</u>
Full generic retpoline, IBPB conditional, IBRS_FW, STIBP conditional, RSB filling
                                                                       Not affected
fpu vme de pse tsc msr pae mce cx8 apic sep mtrr pge mca cmov pat pse36 clflush dts acpi mmx fxsr
                                                                                                                                                                                          TO THE STATE OF
```

- 1. Para el núcleo que se muestra en el Figura 1, y para un programa que implemente la multiplicación de matrices con datos flotantes en doble precisión (use variables globales):
 - 1.1 Modifique el código C para reducir el tiempo de ejecución (evalúe el tiempo y modifique sólo el trozo que hace la multiplicación y el trozo que se muestra en la Figura 1). 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

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

main()
{
    ...
    for (ii=0; ii<40000;ii++) {
        X1=0; X2=0;
        for(i=0; i<5000;i++) X1+=2*s[i].a+ii;
        for(i=0; i<5000;i++) X2+=3*s[i].b-ii;

    if (X1<X2) R[ii]=X1 else R[ii]=X2;
}
    ...
}</pre>
```

A) MULTIPLICACIÓN DE MATRICES:

CAPTURA CÓDIGO FUENTE: pmm-secuencial.c

```
#include <time.h>
#include <stdlib.h>
#include <stdio.h>
#include <math.h>
#define MAX 1024
int A[MAX][MAX], B[MAX][MAX], C[MAX][MAX];
int main(int argc, char** argv){
        struct timespec cgt1, cgt2;
                                                          //Medicion de tiempo
        double ncgt;
        if(argc < 2) {
                fprintf(stderr, "Falta el tamanio de la matriz\n");
                exit(-1);
        unsigned int n = atoi(argv[1]);
        int i, j;
        if(n > MAX){
                n = MAX;
```

```
for(i = 0; i < n; i++){}
                for(j = 0; j < n; j++){
                         A[i][j] = n + (2*j) - i + 1;
                         B[i][j] = n + (2*j) - i + 1;
                         C[i][j] = 0;
                }
        }
        if(n <= 11){
                printf("\nMatriz A:\n");
                for(i = 0; i < n; i++){</pre>
                         for(j = 0; j < n; j++){
                                 printf(" %d ", A[i][j]);
                         printf("\n");
                }
                printf("\nMatriz B:\n");
                 for(i = 0; i < n; i++){
                         for(j = 0; j < n; j++){
                                 printf(" %d ", B[i][j]);
                         printf("\n");
                }
        }
        clock_gettime(CLOCK_REALTIME,&cgt1);
        for(i = 0; i < n; i++) {</pre>
                 for(j = 0; j < n; j++) {
                         for(int k = 0; k < n; k++){
                                 C[i][j] += A[i][k] * B[k][j];
                         }
                }
        clock_gettime(CLOCK_REALTIME, &cgt2);
        ncgt=(double) (cgt2.tv_sec-cgt1.tv_sec) + (double) ((cgt2.tv_nsec-cgt1.tv_nsec) / (1.e+9));
        printf("\nTiempo matriz tamanio %d: %11.9f", n, ncgt);
        if(n <= 11){
                         printf("\nMatriz resultado:\n");
                for(i = 0; i < n; i++){
                         for(j = 0; j < n; j++){
                                 printf(" %d ", C[i][j]);
                         printf("\n");
                }
        }
        return 0;
}
```

1.1. MODIFICACIONES REALIZADAS (al menos dos modificaciones):

Modificación a) –explicación-: Desenrrollado de bucle: 4 operaciones por bucle Modificación b) –explicación-: Trasponer la matriz B para aprovechar la localidad de los accesos

1.1. CÓDIGOS FUENTE MODIFICACIONES

a) Captura de pmm-secuencial-modificado_a.c

```
#include <time.h>
#include <stdlib.h>
#include <stdio.h>
#include <math.h>
#define MAX 1024
int A[MAX][MAX], B[MAX][MAX], C[MAX][MAX];
int main(int argc, char** argv){
        struct timespec cgt1, cgt2;
                                                         //Medicion de tiempo
        double ncgt;
        if(argc < 2) {
                fprintf(stderr, "Falta el tamanio de la matriz\n");
                exit(-1);
        }
        unsigned int n = atoi(argv[1]);
        int i, j;
        if(n > MAX){
                n = MAX;
        }
        for(i = 0; i < n; i++){
                for(j = 0; j < n; j++){
                        A[i][j] = n + (2*j) - i + 1;
                        B[i][j] = n + (2*j) - i + 1;
                        C[i][j] = 0;
                }
        }
        if(n <= 11){
                printf("\nMatriz A:\n");
                for(i = 0; i < n; i++){
                        for(j = 0; j < n; j++){
                                 printf(" %d ", A[i][j]);
                        printf("\n");
                }
                printf("\nMatriz B:\n");
                for(i = 0; i < n; i++){
                        for(j = 0; j < n; j++){
```

```
printf(" %d ", B[i][j]);
                         printf("\n");
                }
        }
        clock_gettime(CLOCK_REALTIME, &cgt1);
        for(i = 0; i < n; i++) {</pre>
                for(j = 0; j < n; j++) {
                        for(int k = 0; k < n; k += 4){
                                 C[i][j] += A[i][k] * B[k][j] + A[i][k+1] * B[k+1]
[j] + A[i][k+2] * B[k+2][j] + A[i][k+3] * B[k+3][j];
                         }
                }
        }
        clock_gettime(CLOCK_REALTIME, &cgt2);
        ncgt=(double) (cgt2.tv_sec-cgt1.tv_sec) + (double) ((cgt2.tv_nsec-
cgt1.tv_nsec) / (1.e+9));
        printf("\nTiempo matriz tamanio %d: %11.9f", n, ncgt);
        if(n <= 11){
                         printf("\nMatriz resultado:\n");
                for(i = 0; i < n; i++){
                        for(j = 0; j < n; j++){
                                 printf(" %d ", C[i][j]);
                         printf("\n");
                }
        }
        return 0;
}
```

```
AC/bp4/ejer1
→ gcc pmm-secuencial-modificado-a.c -o pmm_a

AC/bp4/ejer1
→ ./pmm_a 3

Matriz A:
4 6 8
3 5 7
2 4 6

Matriz B:
4 6 8
3 5 7
2 4 6

Tiempo matriz tamanio 3: 0.000008152

Matriz resultado:
50 86 122
41 71 101
32 56 80
```

b) Captura de pmm-secuencial-modificado_b.c

```
#include <time.h>
#include <stdlib.h>
#include <stdio.h>
#include <math.h>
#define MAX 1024
double A[MAX][MAX], B[MAX][MAX], B_T[MAX][MAX], C[MAX][MAX];
int main(int argc, char** argv){
        struct timespec cgt1, cgt2;
                                                         //Medicion de tiempo
        double ncgt;
        if(argc < 2) {
                fprintf(stderr, "Falta el tamanio de la matriz\n");
                exit(-1);
        }
        unsigned int n = atoi(argv[1]);
        int i, j;
        if(n > MAX){
                n = MAX;
        }
        for(i = 0; i < n; i++){
                for(j = 0; j < n; j++){
                        A[i][j] = n + (2*j) - i + 1;
                        B[i][j] = n + (2*j) - i + 1;
                        C[i][j] = 0;
                }
        }
        if(n <= 11){
                printf("\nMatriz A:\n");
                for(i = 0; i < n; i++){
                        for(j = 0; j < n; j++){
                                 printf(" %d ", A[i][j]);
                        printf("\n");
                }
                printf("\nMatriz B:\n");
                for(i = 0; i < n; i++){
                        for(j = 0; j < n; j++){
                                 printf(" %d ", B[i][j]);
                        printf("\n");
                }
        }
```

```
clock_gettime(CLOCK_REALTIME,&cgt1);
        for(i = 0; i < n; i++){
                for(j = 0; j < n; j++){
                        B_T[i][j] = B[j][i];
                }
        }
        for(i = 0; i < n; i++) {
                for(j = 0; j < n; j++) {
                        for(int k = 0; k < n; k++){
                                 C[i][j] += A[i][k] * B[j][k];
                        }
                }
        }
        clock_gettime(CLOCK_REALTIME, &cgt2);
        ncgt=(double) (cgt2.tv_sec-cgt1.tv_sec) + (double) ((cgt2.tv_nsec-
cgt1.tv_nsec) / (1.e+9));
        printf("\nTiempo matriz tamanio %d: %11.9f", n, ncgt);
        if(n <= 11){
                        printf("\nMatriz resultado:\n");
                for(i = 0; i < n; i++){
                        for(j = 0; j < n; j++){
                                printf(" %d ", C[i][j]);
                        printf("\n");
                }
        }
        return 0;
}
```

```
AC/bp4/ejer1

→ gcc pmm-secuencial-modificado-b.c -o pmm b

AC/bp4/ejer1

→ ./pmm_b 3

Matriz A:

4 6 8

3 5 7

2 4 6

Matriz B:

4 6 8

3 5 7

2 4 6

Tiempo matriz tamanio 3: 0.000040352

Matriz resultado:
50 86 122
41 71 101
32 56 80
```

1.1. TIEMPOS: tamaño matriz 1000 para poder apreciar las diferencias

Modificación	Breve descripción de las modificaciones	-02
Sin modificar	/	4.027477799
Modificación a)	Desenrollado de bucle	3.949253156
Modificación b)	Trasposición de la matriz B	0.623872194

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

La modificación b es notablemente mejor que la a y la no modificada. Esto se debe a que, tan solo con trasponer la matriz b y por como se realiza la multiplicación de matrices aprovechamos mucho mejor la localidad de los accesos, mientras que la primera modificación nos ahorramos solamente un bucle.

B) CÓDIGO FIGURA 1:

CAPTURA CÓDIGO FUENTE: figura1-original.c

```
#include <stdio.h>
#include <stdlib.h>
#include <time.h>
#define TAM_S 5000
#define TAM_R 40000
struct{
        int a;
        int b;
} s[TAM_S];
int R[TAM_R];
int main(){
        int i, ii, X1, X2;
        struct timespec cgt1, cgt2;
                                                           //Medicion de tiempo
        double ncgt;
        for (i = 0; i < TAM_S; i++ ) {</pre>
                 s[i].a = i;
                 s[i].b = 2*i;
        }
        clock_gettime(CLOCK_REALTIME, &cgt1);
        for ( ii = 0; ii < TAM_R; ii++ ) {</pre>
                X1 = 0;
                X2 = 0;
                 for(i=0; i < TAM_S; i++) X1+=2*s[i].a+ii;
                 for(i=0; i < TAM_S; i++) X2+=3*s[i].b-ii;
                 if ( X1 < X2 ) R[ii] = X1; else R[ii] = X2;</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("\nVector R: \n");
printf("R[0]=%d R[39999]=%d\n", R[0], R[TAM_R-1]);
printf("Tiempo: %11.9f\n", ncgt);

return 0;
}
```

1.1. MODIFICACIONES REALIZADAS (al menos dos modificaciones):

Modificación a) –explicación-: reducción de bucles y operador ternario Modificación b) –explicación-: desenrrollado de bucle y operador ternario

1.1. CÓDIGOS FUENTE MODIFICACIONES

a) Captura figura1-modificado_a.c

```
#include <stdio.h>
#include <stdlib.h>
#include <time.h>
#define TAM_S 5000
#define TAM R 40000
struct{
        int a;
        int b;
} s[TAM_S];
int R[TAM_R];
int main(){
        int i, ii, X1, X2;
        int sumatorio;
        struct timespec cgt1, cgt2;
                                                            //Medicion de tiempo
        double ncgt;
        for (i = 0; i < TAM_S; i++ ) {</pre>
                 s[i].a = i;
                 s[i].b = 2*i;
        }
        clock_gettime(CLOCK_REALTIME,&cgt1);
        for ( ii = 0; ii < TAM_R; ii++ ) {</pre>
                 X1 = 0;
                 X2 = 0;
                 for ( i = 0; i < TAM_S; i++ ) {</pre>
                         X1 += s[i].a;
                         X2 += s[i].b;
                 sumatorio = TAM_S * ii;
```

```
X1 = (X1*2) + sumatorio;
X2 = (X2*3) - sumatorio;

R[ii] = X1 < X2 ? X1 : 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("\nvector R: \n");
printf("\nvector R: \n");
printf("R[0]=%d R[39999]=%d\n", R[0], R[TAM_R-1]);
printf("Tiempo: %11.9f\n", ncgt);

return 0;
}</pre>
```

```
AC/bp4/ejer1

→ gcc figural-modificado-a.c -o figl_a

AC/bp4/ejer1

→ ./figl_a

Vector R:
R[0]=24995000 R[39999]=-125010000
Tiempo: 0.474866935
```

b) Captura de figura1-modificado_b.c

```
#include <stdio.h>
#include <stdib.h>
#include <time.h>
#define TAM_R 40000

struct{
    int a;
    int b;
} s[TAM_S];
int R[TAM_R];

int main(){
    int i, ii, X1, X2;

    struct timespec cgt1, cgt2;
    double ncgt;
//Medicion de tiempo
```

```
for (i = 0; i < TAM_S; i++ ) {
                s[i].a = i;
                s[i].b = 2*i;
        }
        clock_gettime(CLOCK_REALTIME,&cgt1);
        for ( ii = 0; ii < TAM_R; ii++ ) {</pre>
                X1 = 0;
                X2 = 0;
                for ( i = 0; i < TAM_S; i += 5 ) {</pre>
                         X1 += 2*s[i].a + ii;
                         X1 += 2*s[i+1].a + ii;
                         X1 += 2*s[i+2].a + ii;
                         X1 += 2*s[i+3].a + ii;
                         X1 += 2*s[i+4].a + ii;
                         X2 += 3*s[i].b - ii;
                         X2 += 3*s[i+1].b - ii;
                         X2 += 3*s[i+2].b - ii;
                         X2 += 3*s[i+3].b - ii;
                         X2 += 3*s[i+4].b - ii;
                R[ii] = X1 < X2 ? X1 : 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("\nVector R: \n");
        printf("R[0]=%d R[39999]=%d\n", R[0], R[TAM_R-1]);
        printf("Tiempo: %11.9f\n", ncgt);
        return 0;
}
```

```
AC/bp4/ejer1

→ gcc figural-modificado-b.c -o figl_b

AC/bp4/ejer1

→ ./figl_b

Vector R:
R[0]=24995000 R[39999]=-125010000
Tiempo: 0.417505936
```

1.1. TIEMPOS:

Modificación	Breve descripción de las modificaciones	-O2
Sin modificar	/	0.230489733
Modificación a)	Reducción de bucles y operador ternario	0.097326707
Modificación b)	Desenrrollado de bucle y operador ternario	0.112365086

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

En este caso la mejor opción es la a: en ambas modificaciones juntamos todo dentro de un sólo bucle for interno, pero funciona mejor llamar al struct el menor número de veces posible acumulando los resultados que ahorrarnos bucles con multiples llamadas dentro.

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 que opera con flotantes de doble precisión 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. Sólo se debe evaluar el tiempo del núcleo DAXPY
 - 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 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>
int main (int argc, char **argv){
        struct timespec cgt1, cgt2;
                                                 //Medicion de tiempo
        double ncgt;
        const int A = 150;
                                                 //Constante arbitraria
        if ( argc < 2 ) {
        fprintf(stderr, "Introduzca un num\n");
        exit(-1);
    }
        int n = atoi(argv[1]);
        int x[n], y[n];
        for (int i = 0; i \le n; i++){
                x[i] = i;
                y[i] = 2*i;
```

	-O0	-Os	-O2	-O3
Tiempos ejec.	0.000002823	0.000000675	0.000003571	0.000000385

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



COMENTARIOS QUE EXPLIQUEN LAS DIFERENCIAS EN ENSAMBLADOR:

- O0: sin optimización alguna.
- Os: se reduce el tamaño del archivo respecto a O2. Bastante parecido a O2.
- O2: se reduce el tamaño del archivo respecto a O0. Se emplean instrucciones más eficientes que las usadas en O0.
- O3: el tamaño del archivo aumenta. Se genera un código más complejo con más llamadas a subrutinas.

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)

daxp	y00.s	daxpy0	Os.s	daxpy	02.s	daxpy(03.s
	.file		.file		.file		.file
	"daxpy.c"		"daxpy.c"		"daxpy.c"		"daxpy.c"
	.text		.text		.text		.text
	.section		.section		.section		.section
	.rodata		.rodata.str1		.rodata.str1.1		.rodata.str1
.LC0:		.1, "aMS",@progbit	ts,1	,"aMS",@progbits	,1	.1, "aMS", @progbi	ts,1
	.string	.LC0:		.LC0:		.LC1:	
	"Introduzca		.string		.string		.string
un num\n"			"Introduzca		"Introduzca un		"Introduzca
.LC2:		un num\n"		num\n"		un num\n"	
	.string	.LC2:		.LC2:		.LC4:	
	•	1.602.	otrina	. 202.	.string	1.204.	otrina
50/17 A/10 H	"\ny[0]: %d,		.string		· ·		.string
y[%d]: %d\n"			"\ny[0]: %d,		"\ny[0]: %d,		"\ny[0]: %d,
.LC3:		y[%d]: %d\n"		y[%d]: %d\n"		y[%d]: %d\n"	
	.string	.LC3:		.LC3:		.LC5:	
	"Tiempo:		.string		.string		.string
%11.9f\n"			"Tiempo:		"Tiempo:		"Tiempo:
	.text	%11.9f\n"		%11.9f\n"	•	%11.9f\n"	
	.globl		.section	,022101 (II	.section		.section
	main		.text.startu		.text.startup,		.text.startu
	.type	p,"ax",@progbits		"ax",@progbits		p, "ax", @progbits	
	main,		.globl		.p2align 4		.p2align 4
@function			main		.globl		.globl
main:			.type		main		main
.LFB6:			main,		.type		.type
	.cfi_startpr	@function	/		main,		main,
00	. 5. 1_5 car cpl	main:		Afunction		@function	
OC				@function		*	
	pushq	.LFB6:		main:		main:	
	%rbp		.cfi_startpr	.LFB22:		.LFB22:	
	.cfi_def_cfa	oc			.cfi_startproc		.cfi_startpr
_offset 16			pushq		pushq	oc	
_	.cfi_offset		%rbp		%rbp		pushq
6, -16	1011_011300		.cfi_def_cfa		.cfi_def_cfa_o		%rbp
0, -10	ma. / a	offoot 10	.cii_uei_cia	ffoot 10	.cri_uer_cra_o		
	movq	_offset 16		ffset 16	s: ss	55	.cfi_def_cfa
	%rsp, %rbp		.cfi_offset		.cfi_offset 6,	_offset 16	
	.cfi_def_cfa	6, -16		-16			.cfi_offset
_register 6			movq		movq	6, -16	
	pushq		%rsp, %rbp		%rsp, %rbp		movq
	%r15		.cfi_def_cfa		.cfi_def_cfa_r		%rsp, %rbp
	pushq	_register 6		egister 6			.cfi_def_cfa
	%r14	_register o	pushq	cgister o	pushq	_register 6	.cri_ucr_cru
			•		•	_register 0	
	pushq		%r15		%r15		pushq
	%r13		pushq		pushq		%r15
	pushq		%r14		%r14		pushq
	%r12		pushq		pushq		%r14
	pushq		%r13		%r13		pushq
	%rbx		pushq		pushq		%r13
	subq		%r12		%r12		pushq
	\$168, %rsp						
			pushq		pushq		%r12
	.cfi_offset		%rbx		%rbx		pushq
15, -24			subq		subq		%rbx
	.cfi_offset		\$72, %rsp		\$72, %rsp		subq
14, -32			.cfi_offset		.cfi_offset		\$72, %rsp
,	.cfi_offset	15, -24		15, -24			.cfi_offset
13, -40		,	.cfi offset	-, =-	.cfi_offset	15, -24	
10, 70	ofi offcot	14 22	1011_0113EL	14 22	.011_011366	10, 27	ofi offcot
40 40	.cfi_offset	14, -32	-e: -cc	14, -32	.e: .ec	14 00	.cfi_offset
12, -48			.cfi_offset	l	.cfi_offset	14, -32	.
	.cfi_offset	13, -40		13, -40			.cfi_offset
3, -56			.cfi_offset		.cfi_offset	13, -40	
	movl	12, -48		12, -48			.cfi_offset
	%edi, -	,	.cfi_offset	,	.cfi_offset 3,	12, -48	
164(%rhn)	/ocu1,	3 -56	.5.1_011500	-56	. 3.1 1 2.1 1 300 3,	,	.cfi_offset
164(%rbp)	mov.c	3, -56	mova	- 30	mova	2 56	.cii_uiiset
	movq		movq		movq	3, -56	
	%rsi, -		%fs:40, %rax		%fs:40, %rax		movq
176(%rbp)			movq		movq		%fs:40, %rax
	movq		%rax, -		%rax, -		movq
	%fs:40, %rax	56(%rbp)	,	56(%rbp)	,		%rax, -
		(/	xorl	(/ - / - / / - / / - /	xorl	56(%rbp)	,
	mova						
	movq %ray					σσ(π. ερ)	vorl
56(%rbp)	movq %rax, -		%eax, %eax decl		%eax, %eax	(30.25)	xorl %eax, %eax

	xorl		%edi		\$1, %edi		cmpl
	%eax, %eax		jg		jle		\$1, %edi
	movq		.L2		.L14		jle
	•				movq		-
	%rsp, %rax		movq		•		.L22
	movq				8(%rsi), %rdi		movq
	%rax, %rbx	stderr(%rip), %rs	Si		movl		8(%rsi),
	movl		leaq		\$10, %edx	%rdi	
	\$150, -		.LC0(%rip),		xorl		movl
152(%rbp)	,	%rdi	(177		%esi, %esi		\$10, %edx
102(701.04)	cmpl	70. 02	call		call		xorl
	\$1, -		fputs@PLT		strtol@PLT		%esi, %esi
164(%rbp)			orl		movslq		call
	jg		\$-1, %edi		%eax, %r12		strtol@PLT
	.L2		call		movq		movslq
	movq		exit@PLT		%rax, %r15		%eax, %r12
	•	.L2:			leag		movq
stderr(%rip),	%ray		movq		15(,%r12,4),		%rax, %r15
stueri (Mi Ip),			•	0/501/	13(,/0112,4),		
	movq	0, -1:	8(%rsi),	%rax			leaq
	%rax, %rcx	%rdi			shrq		15(,%r12,4),
	movl		call		\$4, %rax	%rax	
	\$18, %edx		atoi@PLT		salq		shrq
	movĺ		movslq		\$4, %rax		\$4, %rax
	\$1, %esi		%eax, %r12		subq		salq
	leaq		leaq		%rax, %rsp		\$4, %rax
	•						
0/	.LC0(%rip),	0/	15(,%r12,4),		movq		subq
%rdi		%rax			%rsp, %r14		%rax, %rsp
	call		movq		subq		movq
	fwrite@PLT		%r12, %r14		%rax, %rsp		%rsp, %r14
	movl		shrq		leaq		subq
	\$-1, %edi		\$4, %rax		3(%rsp), %rbx		%rax, %rsp
	call		salq		movq		leaq
	exit@PLT		•		•		·
	exit@FL1		\$4, %rax		%rbx, %r13	04.545	3(%rsp),
.L2:			subq		andq	%rbx	
	movq		%rax, %rsp		\$-4, %rbx		movq
	-176(%rbp),		movq		shrq		%rbx, %r13
%rax			%rsp, %r15		\$2, %r13		andq
	addq		subq		testl		\$-4, %rbx
	\$8, %rax		%rax, %rsp		%r15d, %r15d		shrq
	movq		xorl		js		\$2, %r13
	•				-		
	(%rax), %rax		%eax, %eax		.L3		testl
	movq		leaq		movl		%r15d, %r15d
	%rax, %rdi		3(%rsp),		%r15d, %edx		js
	call	%rbx			xorl		.L3
	atoi@PLT		movq		%eax, %eax		leal
	movl		%rbx, %r13		.p2align 4,,10		1(%r15),
	%eax, -		andq		.p2align 3	%ecx	_(//
140(%rbp)	πεαχ,		•	.L4:	.pzatign 5	70007	cmpl
140 (%) bb)			\$-4, %rbx	4.	11		•
	movl		shrq		leal		\$2, %r15d
	-140(%rbp),		\$2, %r13		(%rax,%rax),		jbe
%eax		.L3:		%ecx			.L15
	movslq		cmpl		movl		movl
	%eax, %rdx		%eax, %r14d		%eax,		%ecx, %edx
	subq		jl	(%r14,%rax,4)	,		movdga
	\$1, %rdx		.L10		movl		.LC0(%rip),
						0/vmm1	. Loo (701 ± μ),
	movq		leal	0(%ecx, (%rbx,	%xmm1	
	%rdx, -	l	(%rax,%rax),	%rax,4)			movdqa
136(%rbp)	_	%edx			movq		.LC2(%rip),
	movslq		movl		%rax, %rcx	%xmm2	
	%eax, %rdx		%eax,		addq		xorl
	movq	(%r15,%rax,4)	•		\$1, %rax		%eax, %eax
	%rdx, -	' ' ' ' ' '	movl		cmpq		shrl
192(%rbp)	,		%edx, (%rbx,		%rdx, %rcx		\$2, %edx
102 (701 DP)	mova	%rax,4)	(/01 DA)				
	movq	/01 ax, 4)	inaa		jne		salq
	\$0, -		incq		. L4		\$4, %rdx
184(%rbp)			%rax		xorl		.p2align
	movslq		jmp		%edi, %edi	4,,10	
	%eax, %rdx		.L3		leaq		.p2align 3
	movq	.L10:			-96(%rbp),	.L5:	-
	%rdx, -		leaq	%rsi			movdqa
			-88(%rbp),		call		%xmm1, %xmm0
208(%rhn)	,		oo(/oi up),		Juli		
208(%rbp)	,	%roi		i .		I .	paddd
208(%rbp)	movq	%rsi		-11	-		0/
	,	%rsi	xorl	clock_gettime@PL1			%xmm2, %xmm1
208(%rbp) 200(%rbp)	movq \$0, -	%rsi	%edi, %edi	clock_gettime@PL1	testl		movups
	movq	%rsi		clock_gettime@PL1			
	movq \$0, -	%rsi	%edi, %edi	clock_gettime@PL1	testl	(%r14,%rax)	movups
	movq \$0, - cltq leaq		%edi, %edi call	clock_gettime@PLT	testl %r15d, %r15d jle	(%r14,%rax)	movups %xmm0,
200(%rbp)	movq \$0, -	%rsi clock_gettime@PLT	%edi, %edi call	clock_gettime@PLT	testl %r15d, %r15d jle .L7	(%r14,%rax)	movups %xmm0, pslld
	movq \$0, - cltq leaq 0(,%rax,4),		%edi, %edi call r xorl	clock_gettime@PL1	testl %r15d, %r15d jle .L7 leal	(%r14,%rax)	movups %xmm0, pslld \$1, %xmm0
200(%rbp)	movq \$0, - cltq leaq		%edi, %edi call	clock_gettime@PL1	testl %r15d, %r15d jle .L7	(%r14,%rax)	movups %xmm0, pslld

	subq		incq		\$1, %eax	(%rbx,%rax)	
	\$1, %rax		%rax		addq		addq
	addq		cmpl		\$2, %rdx		\$16, %rax
	%rdx, %rax		•				· '
	•		%eax, %r14d		.p2align 4,,10		cmpq
	movl		jl		.p2align 3		%rdx, %rax
	\$16, %esi		. L11	.L6:			jne
	movl		imull		imull		.L5
	\$0, %edx		\$150,		\$150,		movl
	divq	(%r15,%rax,4), %		(%r14,%rax,4), %			%ecx, %eax
	%rsi	(*** == , *** ***, **, ***	addl	(10. = 1, 10. 0.11, 1,,	addl		andl
	imulq		%edx, (%rbx,		%ecx, (%rbx,		\$-4, %eax
	\$16, %rax,	%rax,4)		%rax,4)			andl
%rax			jmp		addq		\$3, %ecx
	subq		. L5		\$1, %rax		je
	%rax, %rsp	.L11:			cmpq		.L6
	movq		xorl		%rax, %rdx	.L4:	-
	%rsp, %rax		%edi, %edi		jne	1.54.	movslq
					-		
	addq		leaq		. L6		%eax, %rdx
	\$3, %rax		-72(%rbp),	.L7:			leal
	shrq	%rsi			xorl		(%rax,%rax),
	\$2, %rax		call		%edi, %edi	%ecx	
	salq				leag		movl
	•	clock gettime@DL	т				
	\$2, %rax	clock_gettime@PL		0/roi	-80(%rbp),	(0/r14 0/rdy 4)	%eax,
	movq		movq	%rsi		(%r14,%rdx,4)	_
	%rax, -		-64(%rbp),		call		movl
128(%rbp)		%rax					%ecx, (%rbx,
' '	movl		subq	clock_gettime@PL	Т	%rdx,4)	. , ,
	-140(%rbp),		-80(%rbp),	gottimo@i L	movq	, .,	leal
%02V	740 (101 nh),	%ray	oo(/oi uh),		•		
%eax	_	%rax	_		-72(%rbp),		1(%rax),
	movslq		movl	%rax		%edx	
	%eax, %rdx		%r14d, %edx		pxor		cmpl
	subq		cvtsi2sdq		%xmm0, %xmm0		%edx, %r15d
	\$1, %rdx		%rax, %xmm0		movl		jl
							.L6
	movq		movq		%r15d, %edx		
	%rdx, -		-72(%rbp),		subq		movslq
120(%rbp)		%rax			-88(%rbp),		%edx, %rcx
	movslq		subq	%rax			addl
	%eax, %rdx		-88(%rbp),		pxor		\$2, %eax
		%rax	00(/// 56//		•		movl
	movq	/01 ax			%xmm1, %xmm1		
	%rdx, %r14		divsd		movl		%edx,
	movl		.LC1(%rip),		(%rbx,%r12,4),	(%r14,%rcx,4)	
	\$0, %r15d	%×mm0		%ecx			addl
	movslq		cvtsi2sdq		leaq		%edx, %edx
	%eax, %rdx		%rax, %xmm1		.LC2(%rip),		movl
				04 - 41	.LC2(/01 1p),		
	movq		movl	%rdi			%edx, (%rbx,
	%rdx, %r12		(%rbx,		cvtsi2sdq	%rcx,4)	
	movl	%r12,4), %ecx			%rax, %xmm0		cmpl
	\$0, %r13d		leag		movq		%eax, %r15d
	cltq		.LC2(%rip),		-80(%rbp),		jl
	•	0/mdi	1202(701 1p)/	0/501/	00(701 55))		
	leaq	%rdi	,	%rax			.L7
	0(,%rax,4),		xorl		subq		movslq
%rdx			%eax, %eax		-96(%rbp),		%eax, %rdx
	movl		movl	%rax			movl
	\$16, %eax		0(,%r13,4),		divsd		%eax,
	subq	%esi	(, == / . / /		.LC1(%rip),	(%r14,%rdx,4)	,
	•	70031	added	9/ymmQ	. LOT (701 TH) ,	(/UI 17,/01 UA,4)	addl
	\$1, %rax		addsd	%×mm0			addl
	addq		%xmm1, %xmm0		cvtsi2sdq		%eax, %eax
	%rdx, %rax		movsd		%rax, %xmm1		movl
	movl		%xmm0, -		movl		%eax, (%rbx,
	\$16, %esi	104(%rbp)	,		0(,%r13,4),	%rdx,4)	, , , , ,
	movl	()	call	%esi	- (1/0. =0 -1	.L7:	
				,,,,,,,,,	vorl	'-''	1000
	\$0, %edx		printf@PLT		xorl		leaq
	divq		movsd		%eax, %eax		-96(%rbp),
			101(0/rhn)		addsd	%rsi	
	%rsi		-104(%rbp),				
	%rsi imulq	%×mm0	-104(%TDP),		%xmm1, %xmm0		xorl
	imulq	%×mm0			%xmm1, %xmm0		
%ray		%×mm0	leaq		%xmm1, %xmm0 movsd		%edi, %edi
%rax	imulq \$16, %rax,			104/0/rbs	%xmm1, %xmm0		
%rax	imulq \$16, %rax, subq	%xmm0 %rdi	leaq .LC3(%rip),	104(%rbp)	%xmm1, %xmm0 movsd %xmm0, -		%edi, %edi call
%rax	imulq \$16, %rax,		leaq	104(%rbp)	%xmm1, %xmm0 movsd	clock_gettime@PL	%edi, %edi call
%rax	imulq \$16, %rax, subq		leaq .LC3(%rip),	104(%rbp)	%xmm1, %xmm0 movsd %xmm0, -		%edi, %edi call
%rax	imulq \$16, %rax, subq %rax, %rsp movq		leaq .LC3(%rip), movb \$1, %al	104(%rbp)	<pre>%xmm1, %xmm0 movsd %xmm0, - call printf@PLT</pre>	clock_gettime@PL ⁻	%edi, %edi call
%rax	imulq \$16, %rax, subq %rax, %rsp movq %rsp, %rax		leaq .LC3(%rip), movb \$1, %al call	104(%rbp)	%xmm1, %xmm0 movsd %xmm0, - call printf@PLT movsd	clock_gettime@PL ⁻	%edi, %edi call [
%rax	imulq \$16, %rax, subq %rax, %rsp movq %rsp, %rax addq		leaq .LC3(%rip), movb \$1, %al call printf@PLT	, , ,	<pre>%xmm1, %xmm0 movsd %xmm0, - call printf@PLT</pre>	clock_gettime@PL ⁻ .L8:	%edi, %edi call
%rax	imulq \$16, %rax, subq %rax, %rsp movq %rsp, %rax addq \$3, %rax		leaq .LC3(%rip), movb \$1, %al call printf@PLT movq	104(%rbp) %xmm0	<pre>%xmm1, %xmm0 movsd %xmm0, - call printf@PLT movsd -104(%rbp),</pre>	clock_gettime@PL ⁻	%edi, %edi call r leal -1(%r15),
%rax	imulq \$16, %rax, subq %rax, %rsp movq %rsp, %rax addq		leaq .LC3(%rip), movb \$1, %al call printf@PLT	, , ,	%xmm1, %xmm0 movsd %xmm0, - call printf@PLT movsd	clock_gettime@PL ⁻ .L8:	%edi, %edi call [
%rax	imulq \$16, %rax, subq %rax, %rsp movq %rsp, %rax addq \$3, %rax shrq	%rdi	leaq .LC3(%rip), movb \$1, %al call printf@PLT movq	, , ,	<pre>%xmm1, %xmm0 movsd %xmm0, - call printf@PLT movsd -104(%rbp), movl</pre>	clock_gettime@PL ⁻ .L8:	%edi, %edi call T leal -1(%r15),
%rax	imulq \$16, %rax, subq %rax, %rsp movq %rsp, %rax addq \$3, %rax shrq \$2, %rax		leaq .LC3(%rip), movb \$1, %al call printf@PLT movq -56(%rbp),	, , ,	<pre>%xmm1, %xmm0 movsd %xmm0, - call printf@PLT movsd -104(%rbp), movl \$1, %eax</pre>	clock_gettime@PL ⁻ .L8:	<pre>%edi, %edi call leal -1(%r15), cmpl \$2, %eax</pre>
%rax	imulq \$16, %rax, subq %rax, %rsp movq %rsp, %rax addq \$3, %rax shrq \$2, %rax salq	%rdi	leaq .LC3(%rip), movb \$1, %al call printf@PLT movq -56(%rbp),	, , ,	<pre>%xmm1, %xmm0 movsd %xmm0, - call printf@PLT movsd -104(%rbp), movl \$1, %eax leaq</pre>	clock_gettime@PL ⁻ .L8:	<pre>%edi, %edi call leal -1(%r15), cmpl \$2, %eax jbe</pre>
%rax	imulq \$16, %rax, subq %rax, %rsp movq %rsp, %rax addq \$3, %rax shrq \$2, %rax salq \$2, %rax	%rdi	leaq .LC3(%rip), movb \$1, %al call printf@PLT movq -56(%rbp), xorq %fs:40, %rax	%xmm0	<pre>%xmm1, %xmm0 movsd %xmm0, - call printf@PLT movsd -104(%rbp), movl \$1, %eax</pre>	clock_gettime@PL ⁻ .L8:	<pre>%edi, %edi call leal -1(%r15), cmpl \$2, %eax jbe .L16</pre>
%rax	imulq \$16, %rax, subq %rax, %rsp movq %rsp, %rax addq \$3, %rax shrq \$2, %rax salq	%rdi	leaq .LC3(%rip), movb \$1, %al call printf@PLT movq -56(%rbp), xorq %fs:40, %rax je	, , ,	<pre>%xmm1, %xmm0 movsd %xmm0, - call printf@PLT movsd -104(%rbp), movl \$1, %eax leaq .LC3(%rip),</pre>	clock_gettime@PL ⁻ .L8:	<pre>%edi, %edi call leal -1(%r15), cmpl \$2, %eax jbe .L16 movl</pre>
%rax	imulq \$16, %rax, subq %rax, %rsp movq %rsp, %rax addq \$3, %rax shrq \$2, %rax salq \$2, %rax	%rdi	leaq .LC3(%rip), movb \$1, %al call printf@PLT movq -56(%rbp), xorq %fs:40, %rax	%xmm0	<pre>%xmm1, %xmm0 movsd %xmm0, - call printf@PLT movsd -104(%rbp), movl \$1, %eax leaq</pre>	clock_gettime@PL ⁻ .L8:	<pre>%edi, %edi call leal -1(%r15), cmpl \$2, %eax jbe .L16</pre>

112(%rbp)			call		printf@PLT		movl
	movl				movq		\$4, %eax
	\$0, -	stack_chk_fail	@PLT		-56(%rbp),		shrl
144(%rbp)		.L7:		%rax			\$2, %edx
	jmp		leaq		xorq		salq
	.L3		-40(%rbp),		%fs:40, %rax		\$4, %rdx
.L4:		%rsp			jne		addq
	movq		xorl		.L15		\$4, %rdx
	-128(%rbp),		%eax, %eax		leaq		.p2align
%rax	_		popq		-40(%rbp),	4,,10	
	movl		%rbx	%rsp	_		.p2align 3
	-144(%rbp),		popq		xorl	.L11:	
%edx			%r12		%eax, %eax		movdqu
	movslq		popq		popq	04	(%r14,%rax),
	%edx, %rdx		%r13		%rbx	%xmm1	
	movl		popq		popq		movdqu
9/004	-144(%rbp),		%r14		%r12	0/v/mm2	(%r14,%rax),
%ecx	may.1		popq		popq	%xmm3	mandan
	movl		%r15		%r13		movdqu
0/rdy 4)	%ecx, (%rax,		popq		popq	0/x/mm 4	(%rbx,%rax),
%rdx,4)	movl		%rbp .cfi_def_cfa		%r14	%xmm4	pslld
	-144(%rbp),	7, 8	.cii_uei_cia		popq %r15		\$2, %xmm1
%eax	-144(%i bp),	1, 0	ret		popq		paddd
70CUX	leal		.cfi_endproc		%rbp		%xmm3, %xmm1
	(%rax,%rax),		.cri_cnaproc		.cfi_remember_		movdqa
%ecx	(101 41/101 41/)	.LFE6:	.size	state	.011_10000000		%xmm1, %xmm0
70COX	movq	121201	main,main	State	.cfi_def_cfa		pslld
	-112(%rbp),		.section	7, 8	uoo. u		\$4, %xmm0
%rax	(56//		.rodata.cst8	1, 5	ret		psubd
	movl	, "aM", @progbits,		.L3:			%xmm1, %xmm0
	-144(%rbp),	, , 61 ,	.align 8		.cfi_restore_s		pslld
%edx	(177	.LC1:	•	tate			\$1, %xmm0
	movslq		. long		leaq		paddd
	%edx, %rdx		0		-96(%rbp),		%xmm4, %xmm0
	movl		. long	%rsi			movups
	%ecx, (%rax,		1104006501		xorl		%xmm0,
%rdx,4)			.ident		%edi, %edi	(%rbx,%rax)	
	addl		"GCC: (Arch		call		addq
	\$1, -	Linux 9.3.0-1) 9	.3.0"				\$16, %rax
144(%rbp)			.section	clock_gettime@PL	Т		cmpq
.L3:			.note.GNU-		jmp		%rdx, %rax
	movl	stack,"",@progbi	ts		. L7		jne
	-144(%rbp),			.L15:			.L11
%eax					call		movl
	cmpl						%r15d, %edx
	-140(%rbp),			stack_chk_fail(@PLT		andl
%eax				.L14:			\$-4, %edx
	jle				movq		leal
	. L4			04	stderr(%rip),	04.5	1(%rdx),
	leaq			%rcx	may/1	%eax	omn1
0/rov	-96(%rbp),				movl		cmpl
%rax	mova				\$18, %edx		%edx, %r15d
	movq %rov %roi				movl		je
	%rax, %rsi movl				\$1, %esi leaq	.L10:	.L9
	\$0, %edi				.LCO(%rip),		movslq
	call			%rdi	. LCO(/01 TP),		%eax, %rdx
	Juli			, or u.s.	call		imull
clock_gettime@PL	Т				fwrite@PLT		\$150,
J LOOK_GC CITHICWFL	movl				orl	(%r14,%rdx,4), %	
	\$1, -				\$-1, %edi	(31 ±4/31 ux/4)/ /0	addl
148(%rbp)	1				call		%ecx, (%rbx,
-(,	jmp				exit@PLT	%rdx,4)	, (
	.L5				.cfi_endproc	' ' '	leal
.L6:				.LFE22:			1(%rax),
	movq				.size	%edx	• • • •
	-128(%rbp),				main,main		cmpl
%rax					.section		%r15d, %edx
	movl				.rodata.cst8,"		jg
	-148(%rbp),			aM",@progbits,8	,		.L9
%edx				_ ,	.align 8		movslq
	movslq			.LC1:			%edx, %rdx
	%edx, %rdx				.long		addl
	movl				0		\$2, %eax
	(%rax,				. long		imull
%rdx,4), %eax					1104006501		\$150,
	imull				.ident	(%r14,%rdx,4), %	
	-152(%rbp),				"GCC: (Arch		addl
		1		1		1	

	11. 0.0.1.0.0.0		0/ /0/ . !
%eax	Linux 9.3.0-1) 9.3.0"	0(%ecx, (%rbx,
	.section .note.GNU-	%rdx,4)	amn l
	stack,"",@progbits		cmpl %r15d, %eax
	Stack, , wprognits		
			jg .L9
			cltq
			imull
			\$150,
		(%r14,%rax,4), %	
		(701 14, 701 0.7, 4), 700	addl
			%edx, (%rbx,
		%rax,4)	weux, (wibx,
		.L9:	
		.L9.	xorl
			%edi, %edi
			leaq
		%rsi	-80(%rbp),
		761 51	0011
			call
		-11+	_
		clock_gettime@PL1	
			movq
		0/	-72(%rbp),
		%rax	
			pxor
			%xmm0, %xmm0
			movl
			%r15d, %edx
			subq
			-88(%rbp),
		%rax	
			pxor
			%xmm1, %xmm1
			movl
			(%rbx,
		%r12,4), %ecx	
			leaq
			.LC4(%rip),
		%rdi	
			cvtsi2sdq
			%rax, %xmm0
			movq
			-80(%rbp),
		%rax	` '''
			subq
			-96(%rbp),
		%rax	` ' ' '
			divsd
			.LC3(%rip),
		%×mm0	
		-	cvtsi2sdq
			%rax, %xmm1
			movl
			0(,%r13,4),
		%esi	. (, 25, 1))
			xorl
			%eax, %eax
			addsd
			%xmm1, %xmm0
			movsd
			%xmm0, -
		104(%rbp)	
		237(701 DP)	call
			printf@PLT
			movsd
			-104(%rbp),
		%xmm0	104(%) pb),
		707/111110	movl
			\$1, %eax
			leaq
		0/ rd =	.LC5(%rip),
		%rdi	
			call
			printf@PLT
			movq
			-56(%rbp),
		%rax	
			xorq
ı l		I .	

```
%fs:40, %rax
                  jne
                  .L23
                  leaq
                  -40(%rbp),
%rsp
                  xorl
                  %eax, %eax
                  popq
%rbx
                  popq
                  %r12
                  popq
                  %r13
                  popq
                  %r14
                  popq
                  %r15
                  popq
                  %rbp
                  \verb|.cfi_remembe|
r_state
                  .cfi_def_cfa
7, 8
.L6:
                  .cfi_restore
_state
                  xorl
                  %edi, %edi
                  leaq
-96(%rbp),
%rsi
                  call
clock_gettime@PLT
                  testl
                  %r15d, %r15d
                  jg
                  .L8
                  jmp
                  .L9
.L3:
                  leaq
                  -96(%rbp),
%rsi
                  xorl
                  %edi, %edi
                  call
clock_gettime@PLT
                  jmp
                  .L9
.L16:
                  movl
                  $1, %eax
                  jmp
                  .L10
.L15:
                  xorl
                  %eax, %eax
                  jmp
                  .L4
.L23:
                  call
 _stack_chk_fail@PLT
.L22:
                  movq
stderr(%rip), %rcx
                  movl
                  $18, %edx
                  movl
                  $1, %esi
                  leaq
                  .LC1(%rip),
%rdi
                  call
```

.LFE22: 6,"aM",@prog .LC0:	fwrite@PLT orl \$-1, %edi call exit@PLT .cfi_endproc .size main,main .section .rodata.cst1 jbits,16 .align 16 .long
6,"aM",@prog	\$-1, %edi call exit@PLT .cfi_endproc .size main,main .section .rodata.cst1 jbits,16 .align 16
6,"aM",@prog	call exit@PLT .cfi_endproc .size main,main .section .rodata.cst1 dbits,16 .align 16
6,"aM",@prog	exit@PLT .cfi_endproc .size main,main .section .rodata.cst1 gbits,16 .align 16
6,"aM",@prog	.cfi_endproc .size main,main .section .rodata.cst1 pbits,16 .align 16
6,"aM",@prog	.size main,main .section .rodata.cst1 ubits,16 .align 16
6,"aM",@prog	main,main .section .rodata.cst1 pbits,16 .align 16
	main,main .section .rodata.cst1 pbits,16 .align 16
	.section .rodata.cst1 bits,16 .align 16
	.rodata.cst1 bits,16 .align 16
	bits,16 .align 16
	.align 16
	.align 16
.LCO:	
	. long
	, cong
	0
	. long
	1
	. long
	2
	. long
	3
	.align 16
.LC2:	. a. c. g
	.long
	4
	. long
	4
	. long
	4
	. long
	4
	.section
	.rodata.cst8
,"aM",@progb	ite Q
, an ,eprogr	align 8.
.LC3:	.atiyii o
.LUS.	long
	.long 0
	. long
	1104006501
	.ident
	"GCC: (Arch
Linux 9.3.0-	1) 9.3.0"
	.section
	.note.GNU-
stack,"",@pr	ogbits

	movl		
	%eax, %ecx		
	movq		
	-112(%rbp),		
%rax	-112(%ibp),		
701 ax	movl		
04 - 1	-148(%rbp),		
%edx			
	movslq		
	%edx, %rdx		
	movl		
	(%rax,		
%rdx,4), %eax			
	addl		
	%eax, %ecx		
	movq		
	-112(%rbp),		
%rax			
	movl		
	-148(%rbp),		
%edx	(177		
	movslq		
	%edx, %rdx		
	movl		
	%ecx, (%rax,		
%rdx,4)	,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,		
/01 U/, 4)	addl		
	\$1, -		
148(%rbp)	Ψ±, -		
.L5:			
	movl		
	-148(%rbp),		
%oov	-146(%ibb),		
%eax	cmpl		
	140/0/mbn)		
0/	-140(%rbp),		
%eax			
	jle		
	. L6		
	leaq		
	-80(%rbp),		
%rax			
	movq		
	%rax, %rsi		
	movl		
	\$0, %edi		
	call		
clock_gettime@PL	_T		
	movq		
	-80(%rbp),		
%rdx			
	movq		
	-96(%rbp),		
%rax			
	subq		
	%rax, %rdx		
	movq		
	%rdx, %rax		
	cvtsi2sdq		
	%rax, %xmm1		
	movq		
	-72(%rbp),		
%rdx			
	movq		
	-88(%rbp),		
%rax	X = -12//		
1	subq		
	%rax, %rdx		
	movq		
	%rdx, %rax		
	cvtsi2sdq		
	%rax, %xmm0		
	movsd		
%/vmm2	.LC1(%rip),		
%xmm2	dived		
	divsd		
	%xmm2, %xmm0		
	addsd		
	%xmm1, %xmm0		
	movsd		
	%xmm <u>0</u> , -		71

Cuaderno de prácticas	de Arquitectura	de Computadores.	Grado en	Ingenieria	Informatica