

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): Javier Galera Garrido

Grupo de prácticas:

Fecha de entrega:

Fecha evaluación en clase:

[-RECORDATORIO, quitar todo este texto en rojo del cuaderno definitivo–

1. COMENTARIOS

1) Este cuaderno de prácticas se utilizará para asignarle una puntuación durante la evaluación continua de prácticas y también lo utilizará como material de estudio y repaso para preparar el examen de prácticas escrito. Luego redáctelo con cuidado, y sea ordenado y claro.

2) No use máquinas virtuales. Debe obtener los resultados en su PC (PC del aula o PC personal).

3) Debe modificar el prompt en los computadores que utilice en este bloque práctico para que aparezca su nombre y apellidos, su usuario (\u), el computador (\h), el directorio de trabajo del bloque práctico (\w), la fecha (\D) completa (%F) y el día (%A) . Para modificar el prompt utilice lo siguiente (si es necesario, use export delante):

```
PS1="[NombreApellidos \u@\h:\w] \D{%F %A}\n$"
```

donde NombreApellidos es su nombre seguido de sus apellidos, por ejemplo: Juan Ortuño Vilariño

2. NORMAS SOBRE EL USO DE LA PLANTILLA

1) Usar **interlineado SENCILLO**.

2) Respetar los tipos de letra y tamaños indicados:

- Calibri-11 o Liberation Serif-11 para el texto

- **Courier New-10 o Liberation Mono-10 para nombres de fichero, comandos, variables de entorno, etc., cuando se usan en el texto.**

3) Insertar las capturas de pantalla donde se pidan y donde se considere oportuno. En particular, los listados de código se deben insertar como capturas de pantalla. En todas las capturas de pantalla, incluidas las de los listados de código, debe aparecer el directorio y usuario. El tamaño de letra en las capturas debe ser similar al tamaño que se está usando en el texto.

Recuerde que debe adjuntar al zip de entrega, el pdf de este fichero, todos los ficheros con código fuente implementados/utilizados y el resto de ficheros que haya implementado/utilizado (scripts, hojas de cálculo, etc.)]

Denominación de marca del chip de procesamiento o procesador (se encuentra en /proc/cpuinfo): *Intel(R) Core(TM) i5-7200U CPU @ 2.50GHz*

Sistema operativo utilizado: *Ubuntu 18.04LTS*

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

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

1. Para el núcleo que se muestra en el Figura 1, y para un programa que implemente la multiplicación de matrices (use variables globales):

1.1 Modifique el código C para reducir el tiempo de ejecución del mismo. Justifique los tiempos obtenidos (use -O2) a partir de la modificación realizada. Incorpore los códigos modificados en el cuaderno.

1.2 Genere los códigos en ensamblador con -O2 para el original y dos códigos modificados obtenidos en el punto anterior (incluido el que supone menor tiempo de ejecución) e incorpórellos 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;
    }
    ...
}
```

A) MULTIPLICACIÓN DE MATRICES:

CAPTURA CÓDIGO FUENTE: pmm-secuencial.c

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

#define MAX 3000

int a[MAX][MAX], b[MAX][MAX], c[MAX][MAX];

void imprimirMatriz(int m[MAX][MAX], int tam){
    int i, j;
    for (i=0; i<tam; i++) {
        for (j=0; j<tam; j++){
            printf("%d ", m[i][j]);
        }
        printf("\n");
    }
    printf("\n");
}

int main(int argc, char const *argv[])
{
    // COMPROBACION DE ARGUMENTOS
    if (argc < 2) {
        fprintf(stderr, "Falta el tamaño de la matriz\n");
        exit(-1);
    }

    int N = atoi(argv[1]);
    if(N>MAX) N = MAX;

    int i, j, k;
    struct timespec cgt1, cgt2; double ncgt;

    //Inicializacion
    for (i=0; i<N; i++){
        for (j=0; j<N; j++){
            a[i][j] = 0;
        }
    }
```

```

        b[i][j] = i+j+1;
        c[i][j] = j+(i+2);
    }
}
/*
 * De tamaño 3, por ejemplo:
 * b= 1 2 3
 *     2 3 4
 *     3 4 5
 *
 * c= 2 3 4
 *     3 4 5
 *     4 5 6
 *
 * a= 20 26 32
 *     29 38 47
 *     38 50 62
 */
printf("c[0][0] = %d, c[N][N] = %d\n",c[0][0],c[N-1][N-1]);

//imprimirMatriz(b, N);
//imprimirMatriz(c, N);

clock_gettime(CLOCK_REALTIME,&cgt1);
//Calculo
for (i=0; i<N; i++)
    for (j=0; j<N; j++)
        for (k=0; k<N; k++){
            a[i][j] += b[i][k]*c[k][j];
        }

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

//Solucion
//imprimirMatriz(a,N);
printf("\nA[0][0]=%d\nA[N-1][N-1]=%d\n",a[0][0],a[N-1][N-1]);
printf("\nTiempo (seg.) = %11.9f\n", ncgt);

return 0;
}

```

1.1. MODIFICACIONES REALIZADAS (al menos dos modificaciones):

Modificación a) –explicación–: En esta modificación he calculado primeramente la matriz traspuesta de la matriz C para luego acceder por filas en vez de por columnas, ya que así nos ahorramos, en cada acceso a los valores de esta matriz, fallos de caché porque no están en posiciones contiguas de memoria. Al transponer la matriz y acceder por filas, se puede guardar en caché varios elementos de la fila evitando en cada acceso el fallo de caché.

Modificación b) –explicación–: Una vez aplicada la matriz traspuesta, hacer desenrollado de bucle de 4 en 4 saltos.

1.1. CÓDIGOS FUENTE MODIFICACIONES

a) Captura de pmm-secuencial-modificado_a.c

```

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

#define MAX 3000

int a[MAX][MAX], b[MAX][MAX], c[MAX][MAX];

void imprimirMatriz(int m[MAX][MAX], int tam){
    int i,j;
    for (i=0; i<tam; i++){
        for (j=0; j<tam; j++){
            printf("%d ", m[i][j]);
        }
        printf("\n");
    }
    printf("\n");
}

int main(int argc, char const *argv[])
{

```

```

// COMPROBACION DE ARGUMENTOS
if (argc < 2) {
    fprintf(stderr, "Falta el tamaño de la matriz\n");
    exit(-1);
}

int N = atoi(argv[1]);
if(N>MAX) N = MAX;
int i, j,k;
struct timespec cgt1,cgt2, cgt3; double ncgt,ncgt2;
//Inicializacion
for (i=0; i<N; i++){
    for (j=0; j<N; j++){
        a[i][j] = 0;
        b[i][j] = i+j+1;
        c[i][j] = j+(i+2);
    }
}

//imprimirMatriz(b, N);
//imprimirMatriz(c, N);

clock_gettime(CLOCK_REALTIME,&cgt3);
//Trasponemos la matriz para que el acceso a memoria sea por filas y no
por columnas
for(i=0; i<N; i++){
    for(j=0; j<N; j++){
        int tmp = c[i][j];
        c[i][j] = c[j][i];
        c[j][i] = tmp;
    }
}

clock_gettime(CLOCK_REALTIME,&cgt1);

//Calculo
for (i=0; i<N; i++)
    for (j=0; j<N; j++)
        for (k=0; k<N; k++)
            a[i][j] += b[i][k]*c[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));
ncgt2=(double) (cgt2.tv_sec-cgt3.tv_sec)+( double) ((cgt2.tv_nsec-
cgt3.tv_nsec)/(1.e+9));

//Solucion
//imprimirMatriz(a,N);
printf("\nA[0][0]=%d\nA[N-1][N-1]=%d\n",a[0][0],a[N-1][N-1]);
printf("\nTiempo (seg.) solo multiplicación = %11.9f\n", ncgt);

printf("\nTiempo (seg.) con trasposicion de matriz = %11.9f\t%11.9f
segundos mas\n", ncgt2, ncgt2-ncgt);

return 0;
}

```

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

```

~/Escritorio/segundo/ac/practicas/practica5 4:15:53
$ gcc -O2 pmm-secuencial.c -o pmm-secuencial
~/Escritorio/segundo/ac/practicas/practica5 4:17:45
$ gcc -O2 pmm-modificado.c -o pmm-modificado
~/Escritorio/segundo/ac/practicas/practica5 4:18:00
$ ./pmm-secuencial 5
c[0][0] = 2, c[N][N] = 10
A[0][0]=70
A[N-1][N-1]=290
Tiempo (seg.) = 0.000002192
~/Escritorio/segundo/ac/practicas/practica5 4:18:09
$ ./pmm-modificado 5
A[0][0]=70
A[N-1][N-1]=290
Tiempo (seg.) solo multiplicacion = 0.000000925
Tiempo (seg.) con trasposicion de matriz = 0.000001403 0.000000478 segundos mas
~/Escritorio/segundo/ac/practicas/practica5 4:18:19

```

b) Captura de pmm-secuencial-modificado_b.c

```

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

#define MAX 3000

int a[MAX][MAX], b[MAX][MAX], c[MAX][MAX];

void imprimirMatriz(int m[MAX][MAX], int tam){
    int i,j;
    for (i=0; i<tam; i++){
        for (j=0; j<tam; j++){
            printf("%d ", m[i][j]);
        }
        printf("\n");
    }
    printf("\n");
}

int main(int argc, char const *argv[])
{
    // COMPROBACION DE ARGUMENTOS
    if (argc < 2) {
        fprintf(stderr, "Falta el tamaño de la matriz\n");
        exit(-1);
    }

    int N = atoi(argv[1]);
    if(N>MAX) N = MAX;

    int i, j,k;

    struct timespec cgt1,cgt2, cgt3; double ncgt,ncgt2;
    //Inicializacion
    for (i=0; i<N; i++){
        for (j=0; j<N; j++){
            a[i][j] = 0;
            b[i][j] = i+j+1;
            c[i][j] = j+(i+2);
        }
    }

    //imprimirMatriz(b, N);
    //imprimirMatriz(c, N);

```

```

clock_gettime(CLOCK_REALTIME,&cgt3);
//Trasponemos la matriz para que el acceso a memoria sea por filas y no
por columnas
for(i=0; i<N; i++){
    for(j=0; j<N; j++){
        int tmp = c[i][j];
        c[i][j] = c[j][i];
        c[j][i] = tmp;
    }
}

clock_gettime(CLOCK_REALTIME,&cgt1);
int ultima = N/4, l;
//Calculo
for (i=0; i<N; i++)
    for (j=0; j<N; j++) {
        for (k=0, l=0; l<ultima; k+=4, l++){
            a[i][j] += b[i][k]*c[j][k];
            a[i][j] += b[i][k+1]*c[j][k+1];
            a[i][j] += b[i][k+2]*c[j][k+2];
            a[i][j] += b[i][k+3]*c[j][k+3];
        }
        for(k=ultima*4; k<N; k++){
            a[i][j] += b[i][k]*c[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));
ncgt2=(double) (cgt2.tv_sec-cgt3.tv_sec)+( double) ((cgt2.tv_nsec-
cgt3.tv_nsec)/(1.e+9));

//Solucion
//imprimirMatriz(a, N);
printf("\nA[0][0]=%d\nA[N-1][N-1]=%d\n", a[0][0], a[N-1][N-1]);
printf("\nTiempo (seg.) solo multiplicacion = %11.9f\n", ncgt);

printf("\nTiempo (seg.) con trasposicion de matriz = %11.9f\t%11.9f
segundos mas\n", ncgt2, ncgt2-ncgt);

return 0;
}

```

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

```

~/Escritorio/segundo/ac/practicas/practica5 @ 4:23:19
$ gcc -O2 pmm-modificado_2.c -o pmm-modificado_2
~/Escritorio/segundo/ac/practicas/practica5 @ 4:23:41
$ ./pmm-secuencial 5
c[0][0] = 2, c[N][N] = 10
A[0][0]=70
A[N-1][N-1]=290
Tiempo (seg.) = 0.000001410
~/Escritorio/segundo/ac/practicas/practica5 @ 4:23:53
$ ./pmm-modificado_2 5
A[0][0]=70
A[N-1][N-1]=290
Tiempo (seg.) solo multiplicacion = 0.000000963
Tiempo (seg.) con trasposicion de matriz = 0.000001542 0.000000579 segundos mas

```

1.1. TIEMPOS: para matrices de tamaño 1500 x 1500

Modificación		-O0	-O1	-O2
Sin modificar		21.33872346	4.992951580	4.863861247
Modificación a)		13.843146801	2.896383608	3.163519620
Modificación b)		13.184861515	2.188715671	2.221235431

1.1. COMENTARIOS SOBRE LOS RESULTADOS:

Como podemos observar en los tiempos de ejecución, el programa para una matriz de tamaño 1500 sin optimizaciones da un resultado de 4.8 segundos. Modificando los accesos a memoria caché (trasponer la matriz), el tiempo disminuye 1 segundo, por tanto el cambio es más que notable. Para la optimización 2, en la que se realiza un desenrollado de bucle para 4 elementos, se optimiza el código en ensamblador y disminuye 1 segundo el tiempo de la segunda modificación.

Viendo la tabla en general, se puede observar como se puede disminuir el tiempo de ejecución de 21.33 segundos a 2 segundos optimizando el código y usando una optimización O2 en el compilador.

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

pmm-secuencial.s	pmm-secuencial-modificado_b.s	pmm-secuencial-modificado_c.s
<pre> .file "pmm-secuencial.c" .text .comm a, 36000000, 32 .comm b, 36000000, 32 .comm c, 36000000, 32 .section .rodata .LC0: .string "%d " .text .globl imprimirMatriz .type imprimirMatriz, @function .LFB5: .cfi_startproc pushq %rbp .cfi_def_cfa_of fset 16 -16 gister 6 subq \$32, %rsp movq %rdi, -24(%rbp) movl %esi, -28(%rbp) movl \$0, -8(%rbp) jmp .L2 .L5: movl \$0, -4(%rbp) jmp .L3 .L4: movl -8(%rbp), %eax cltq imulq \$12000, %rax, 24(%rbp) movl %esi, - </pre>	<pre> .file "pmm-modificado.c" .text .comm a, 36000000, 32 .comm b, 36000000, 32 .comm c, 36000000, 32 .section .rodata .LC0: .string "%d " .text .globl imprimirMatriz .type imprimirMatriz, @function imprimirMatriz: .LFB5: .cfi_startproc pushq %rbp .cfi_def_cfa_of fset 16 -16 gister 6 subq \$32, %rsp movq %rdi, -24(%rbp) movl %esi, -28(%rbp) movl \$0, -8(%rbp) jmp .L2 .L5: movl \$0, -4(%rbp) jmp .L3 .L4: movl -8(%rbp), %eax cltq imulq \$12000, %rax, %rdx movq -24(%rbp), %rax addq </pre>	<pre> .file "pmm-modificado_2.c" .text .comm a, 36000000, 32 .comm b, 36000000, 32 .comm c, 36000000, 32 .section .rodata .LC0: .string "%d " .text .globl imprimirMatriz .type imprimirMatriz, @function imprimirMatriz: .LFB5: .cfi_startproc pushq %rbp .cfi_def_cfa_of fset 16 -16 gister 6 subq \$32, %rsp movq %rdi, -24(%rbp) movl %esi, -28(%rbp) movl \$0, -8(%rbp) jmp .L2 .L5: movl \$0, -4(%rbp) jmp .L3 .L4: movl -8(%rbp), %eax cltq imulq \$12000, %rax, %rdx movq -24(%rbp), %rax addq </pre>

28(%rbp)	movl \$0, -		%rax, %rdx movl -4(%rbp), %eax cltq movl (%rdx,%rax,4),		%rax, %rdx movl -4(%rbp), %eax cltq movl (%rdx,%rax,4),
8(%rbp)	jmp .L2	%eax	movl %eax, %esi leaq .LC0(%rip),	%eax	movl %eax, %esi leaq .LC0(%rip),
.L5:	movl \$0, -	%rdi	movl \$0, %eax call printf@PLT addl \$1, -4(%rbp)	%rdi	movl \$0, %eax call printf@PLT addl \$1, -4(%rbp)
4(%rbp)	jmp .L3		movl -8(%rbp), %eax cmpl -28(%rbp), %eax jl .L4 movl \$10, %edi call putchar@PLT addl \$1, -8(%rbp)		movl -8(%rbp), %eax cmpl -28(%rbp), %eax jl .L4 movl \$10, %edi call putchar@PLT addl \$1, -8(%rbp)
.L4:	movl -8(%rbp),	.L3:	movl -8(%rbp), %eax cmpl -28(%rbp), %eax jl .L5 movl \$10, %edi call putchar@PLT nop leave .cfi_def_cfa 7,	.L3:	movl -8(%rbp), %eax cmpl -28(%rbp), %eax jl .L5 movl \$10, %edi call putchar@PLT nop leave .cfi_def_cfa 7,
%eax	cltq imulq \$12000,		ret .cfi_endproc		ret .cfi_endproc
%rax, %rdx	movq -24(%rbp),		.size imprimirMatriz,		.size imprimirMatriz,
%rax	addq %rax, %rdx movl -4(%rbp),	.L2:	.section .rodata .align 8		.section .rodata .align 8
%eax	cltq movl (%rdx,		.string "Falta el tamaño de la matriz\n"		.string "Falta el tamaño de la matriz\n"
%rax,4), %eax	movl %eax, %esi leaq .LC0(%rip),	8	.string "\nA[0][0]=%d\ nA[N-1][N-1]=%d\ n"		.string "\nA[0][0]=%d\ nA[N-1][N-1]=%d\ n"
%rdi	movl \$0, %eax call printf@PLT addl \$1, -	.LFE5:	.string "\nTiempo (seg.) solo multiplicacion = %11.9f\n"		.string "\nTiempo (seg.) solo multiplicacion = %11.9f\n"
4(%rbp)	movl -4(%rbp),	.LC1:	.string "\nTiempo (seg.) con trasposicion de matriz = %11.9f\t%11.9f segundos mas\n"		.string "\nTiempo (seg.) con trasposicion de matriz = %11.9f\t%11.9f segundos mas\n"
%eax	cmpl -28(%rbp),	.LC3:	.text .globl main .type main, @function		.text .globl main .type main, @function
%eax	jl .L4 movl \$10, %edi call putchar@PLT addl \$1, -	.LC4:	.cfi_startproc pushq %rbp .cfi_def_cfa_of		.cfi_startproc pushq %rbp .cfi_def_cfa_of
8(%rbp)	movl -8(%rbp),	.LC5:	.cfi_offset 6, -16		.cfi_offset 6, -16
.L2:	cmpl -28(%rbp),		movq %rsp, %rbp .cfi_def_cfa_re		movq %rsp, %rbp .cfi_def_cfa_re
%eax	jl .L5 movl \$10, %edi call putchar@PLT nop leave .cfi_def_cf	fset 16	subq \$144, %rsp movl %edi, -	fset 16	subq \$144, %rsp movl %edi, -
%eax	ret .cfi_endpro	-16	movq %rsi, -	-16	movq %rsi, -
a 7, 8		gister 6	movq %fs:40, %rax movq %rax, -8(%rbp) xorl %eax, %eax cmpl \$1, -116(%rbp) jg .L7 movq	gister 6	movq %fs:40, %rax movq %rax, -8(%rbp) xorl %eax, %eax cmpl \$1, -116(%rbp) jg .L7 movq
c		116(%rbp)		116(%rbp)	
.LFE5:	.size	128(%rbp)		128(%rbp)	
imprimirMatriz, .-					
imprimirMatriz	.section .rodata .align 8				

<pre> .LC1: .string "Falta el tamaño de la matriz\n" .LC2: .string "c[0][0] = %d, c[N][N] = %d\n" .LC4: .string "\nA[0][0]= %d\nA[N-1][N-1]=%d\n" .LC5: .string "\nTiempo (seg.) = %11.9f\n" .text .globl main .type main, @function main: .LFB6: .cfi_startp roc pushq %rbp .cfi_def_cf a_offset 16 .cfi_offset 6, -16 movq %rsp, %rbp .cfi_def_cf a_register 6 subq \$112, %rsp movl %edi, - 84(%rbp) movq %rsi, - 96(%rbp) movq %fs:40, %rax movq %rax, - 8(%rbp) xorl %eax, %eax cmpl \$1, - 84(%rbp) jg .L7 movq stderr(%rip), %rax movq %rax, %rcx movl \$30, %edx movl \$1, %esi leaq .LC1(%rip), %rdi call fwrite@PLT movl \$-1, %edi call exit@PLT .L7: movq -96(%rbp), %rax addq \$8, %rax movq (%rax), %rax movq %rax, %rdi call atoi@PLT movl %eax, - 100(%rbp) cmpl \$3000, - 108(%rbp) jle .L8 movl \$3000, - 108(%rbp) .L8: movl \$0, -96(%rbp) jmp .L9 .L12: movl \$0, -92(%rbp) jmp .L10 .L11: movl -92(%rbp), %eax cltq movl -96(%rbp), %edx movslq %edx, %rdx imulq \$3000, %rdx, %rdx addq %rdx, %rax leaq 0(,%rax,4), %rdx leaq a(%rip), %rax movl \$0, (%rdx,%rax) movl -96(%rbp), %edx movl -92(%rbp), %eax addl %edx, %eax leal 1(%rax), %ecx movl -92(%rbp), %eax cltq movl -96(%rbp), %edx movslq %edx, %rdx imulq \$3000, %rdx, %rdx addq %rdx, %rax leaq 0(,%rax,4), %rdx leaq b(%rip), %rax movl \$ecx, (%rdx, %rax) movl -96(%rbp), %eax leal 2(%rax), %edx movl -92(%rbp), %eax leal (%rdx,%rax), %ecx movl -92(%rbp), %eax cltq movl -96(%rbp), %edx movslq </pre>	<pre> stderr(%rip), %rax movq %rax, %rcx movl \$30, %edx movl \$1, %esi leaq .LC1(%rip), %rdi call fwrite@PLT movl \$-1, %edi call exit@PLT .L7: movq -128(%rbp), %rax addq \$8, %rax movq (%rax), %rax movq %rax, %rdi call atoi@PLT movl %eax, - 100(%rbp) cmpl \$3000, - 108(%rbp) jle .L8 movl \$3000, - 108(%rbp) .L8: movl \$0, -96(%rbp) jmp .L9 .L12: movl \$0, -92(%rbp) jmp .L10 .L11: movl -92(%rbp), %eax cltq movl -96(%rbp), %edx movslq %edx, %rdx imulq \$3000, %rdx, %rdx addq %rdx, %rax leaq 0(,%rax,4), %rdx leaq a(%rip), %rax movl \$0, (%rdx,%rax) movl -96(%rbp), %edx movl -92(%rbp), %eax addl %edx, %eax leal 1(%rax), %ecx movl -92(%rbp), %eax cltq movl -96(%rbp), %edx movslq %edx, %rdx imulq \$3000, %rdx, %rdx addq %rdx, %rax leaq 0(,%rax,4), %rdx leaq b(%rip), %rax movl \$ecx, (%rdx, %rax) movl -96(%rbp), %eax leal 2(%rax), %edx movl -92(%rbp), %eax leal (%rdx,%rax), %ecx movl -92(%rbp), %eax cltq movl -96(%rbp), %edx movslq </pre>	<pre> stderr(%rip), %rax movq %rax, %rcx movl \$30, %edx movl \$1, %esi leaq .LC1(%rip), %rdi call fwrite@PLT movl \$-1, %edi call exit@PLT .L7: movq -128(%rbp), %rax addq \$8, %rax movq (%rax), %rax movq %rax, %rdi call atoi@PLT movl %eax, - 108(%rbp) cmpl \$3000, - 108(%rbp) jle .L8 movl \$3000, - 108(%rbp) .L8: movl \$0, -104(%rbp) jmp .L9 .L12: movl \$0, -100(%rbp) jmp .L10 .L11: movl -100(%rbp), %eax cltq movl -104(%rbp), %edx movslq %edx, %rdx imulq \$3000, %rdx, %rdx addq %rdx, %rax leaq 0(,%rax,4), %rdx leaq a(%rip), %rax movl \$0, (%rdx,%rax) movl -104(%rbp), %edx movl -100(%rbp), %eax addl %edx, %eax leal 1(%rax), %ecx movl -100(%rbp), %eax cltq movl -104(%rbp), %edx movslq %edx, %rdx imulq \$3000, %rdx, %rdx addq %rdx, %rax leaq 0(,%rax,4), %rdx leaq b(%rip), %rax movl \$ecx, (%rdx, %ecx) movl -104(%rbp), %eax leal 2(%rax), %edx movl -100(%rbp), %eax leal </pre>
--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------	---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------	-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

72(%rbp)	movl %eax, -	%rdx	%edx, %rdx imulq \$3000, %rdx,	%ecx	(%rdx,%rax),
72(%rbp)	cmpl \$3000, -	%rdx	addq %rdx, %rax leaq 0(,%rax,4),	%eax	movl -100(%rbp),
72(%rbp)	jle .L8 movl \$3000, -	%rdx	leaq c(%rip), %rax movl %ecx, (%rdx,	%edx	cltq movl -104(%rbp),
.L8:	movl \$0, -	%rax)	addl \$1, -92(%rbp)	%rdx	movslq %edx, %rdx imulq \$3000, %rdx,
68(%rbp)	jmp .L9	.L10:	movl -92(%rbp), %eax cmpl -100(%rbp),	%rdx	addq %rdx, %rax leaq 0(,%rax,4),
.L12:	movl \$0, -	%eax	j1 .L11 addl \$1, -96(%rbp)	%rax)	leaq c(%rip), %rax movl %ecx, (%rdx,
64(%rbp)	jmp .L10	.L9:	movl -96(%rbp), %eax cmpl -100(%rbp),	.L10:	addl \$1, -100(%rbp)
.L11:	movl -64(%rbp),	%eax	j1 .L12 leaq -32(%rbp), %rax movq %rax, %rsi movl \$0, %edi call	%eax	movl -100(%rbp),
%eax	cltq movl -68(%rbp),		clock_gettime@PLT movl \$0, -96(%rbp) jmp .L13	%eax	cmpl -108(%rbp),
%edx	movslq %edx, %rdx imulq \$3000,	.L16:	movl \$0, -92(%rbp) jmp .L14		j1 .L11 addl \$1, -104(%rbp)
%rdx, %rdx	addq %rdx, %rax leaq 0(,%rax,4),	.L15:	movl -92(%rbp), %eax cltq movl -96(%rbp), %edx movslq %edx, %rdx imulq \$3000, %rdx,	clock_gettime@PLT movl \$0, -104(%rbp) jmp .L13	movl -104(%rbp),
%rdx	leaq a(%rip),		addq %rdx, %rax leaq 0(,%rax,4),	.L16:	cmpl \$0, -108(%rbp),
%rax	movl \$0, (%rdx,	%rdx	leaq c(%rip), %rax movl (%rdx,%rax),	.L15:	j1 .L12 leaq -32(%rbp), %rax movq %rax, %rsi movl \$0, %edi call
%rax)	movl -68(%rbp),	%rdx	movl %eax, -84(%rbp) movl -96(%rbp), %eax cltq movl -92(%rbp), %edx movslq %edx, %rdx imulq \$3000, %rdx,		movl \$0, -100(%rbp) jmp .L14
%edx	movl -64(%rbp),	%rdx	addq %rdx, %rax leaq 0(,%rax,4),	%eax	movl -100(%rbp),
%eax	addl %edx, %eax leal 1(%rax),	%eax	leaq c(%rip), %rax movl (%rdx,%rax),	%edx	cltq movl -104(%rbp),
%ecx	movl -64(%rbp),		movl %eax, -84(%rbp) movl -96(%rbp), %eax cltq movl -92(%rbp), %edx movslq %edx, %rdx imulq \$3000, %rdx,	%rdx	movslq %edx, %rdx imulq \$3000, %rdx,
%eax	cltq movl -68(%rbp),	%rdx	addq %rdx, %rax leaq 0(,%rax,4),	%rdx	addq %rdx, %rax leaq 0(,%rax,4),
%edx	movslq %edx, %rdx imulq \$3000,	%rdx	leaq c(%rip), %rax movl (%rdx,%rax),	%eax	leaq c(%rip), %rax movl (%rdx,%rax),
%rdx, %rdx	addq %rdx, %rax leaq 0(,%rax,4),	%edx	movl -92(%rbp), %eax cltq movl -96(%rbp), %ecx movslq %ecx, %rcx imulq \$3000, %rcx,	%rdx	movl %eax, -84(%rbp) movl -104(%rbp),
%rdx	leaq b(%rip),		addq %rcx, %rax leaq 0(,%rax,4),	%rdx	cltq movl -100(%rbp),
%rax	movl %ecx,	%rcx	leaq c(%rip), %rax movl	%edx	movslq %edx, %rdx imulq \$3000, %rdx,
(%rdx,%rax)	movl -68(%rbp),			%rdx	addq %rdx, %rax leaq 0(,%rax,4),
%eax	leal 2(%rax),	%rcx		%rdx	leaq c(%rip), %rax movl (%rdx,%rax),
%edx	movl			%edx	movl -100(%rbp),

%eax	-64(%rbp),	%rax)	%edx, (%rcx,	%eax	cltq
%rax), %ecx	leal		movl	%ecx	movl
%eax	(%rdx,		-96(%rbp), %eax		-104(%rbp),
%edx	movl	%rdx	cltq		movslq
%rdx, %rdx	-64(%rbp),		movl	%ecx, %rcx	%ecx, %rcx,
%rdx	cltq		-92(%rbp), %edx	%rcx	\$3000, %rcx,
%rax	movl	%rcx	movslq	%rcx	addq
(%rdx, %rax)	-68(%rbp),		%edx, %rdx	%rcx	%rcx, %rax
64(%rbp)	imulq	%rax)	\$3000, %rdx,	%eax	leaq
.L10:	\$3000,	.L14:	addq	%edx	0(, %rax, 4),
%eax	addq	%eax	%rdx, %rax	%rdx	leaq
%eax	leaq		0(, %rax, 4),	%rcx	c(%rip), %rax
68(%rbp)	leaq	.L13:	leaq	%edx, (%rcx,	movl
.L9:	c(%rip),		c(%rip), %rax	%eax	%edx, (%rcx,
%eax	movl	%eax	movl	%rdx	-104(%rbp),
%eax	%ecx,		-84(%rbp), %edx	%ecx	cltq
%eax	addl		movl	%edx	movl
%eax	\$1, -		-100(%rbp),	%rdx	-100(%rbp),
%eax	movl		addl	%rcx	movslq
%eax	-64(%rbp),		\$1, -92(%rbp)	%rdx	%edx, %rdx,
%eax	cmpl		movl	%rcx	imulq
%eax	-72(%rbp),		-92(%rbp), %eax	%rdx	\$3000, %rdx,
%eax	j1		cmpl	%rcx	addq
%eax	.L11		-100(%rbp),	%rcx	%rdx, %rax
%eax	addl		j1	%rdx	leaq
%eax	\$1, -		.L16	%rcx	0(, %rax, 4),
%eax	clock_gettime@PLT		leaq	%rcx	leaq
%eax	.L9:		-64(%rbp), %rax	%rdx	c(%rip), %rax
%eax	movl		movq	%rcx	movl
%eax	-68(%rbp),		%rax, %rsi	%rdx	-84(%rbp), %edx
%eax	cmpl		movl	%rcx	movl
%eax	-72(%rbp),		\$0, %edi	%rdx	%edx, (%rcx,
%eax	j1		call	%rcx	addl
%eax	.L12		movl	%rdx	\$1, -100(%rbp)
%eax	movl		\$0, -96(%rbp)	%rcx	movl
%eax	-72(%rbp),		jmp	%rdx	-100(%rbp)
%eax	leal		.L17	%rcx	addl
%edx	-1(%rax),		movl	%rdx	\$1, -104(%rbp)
%edx	movl		\$0, -92(%rbp)	%rcx	movl
%edx	-72(%rbp),		jmp	%rdx	-104(%rbp),
%edx	leal		.L18	%rcx	cmpl
%edx	-1(%rax),		movl	%rdx	-108(%rbp),
%edx	movl		\$0, -88(%rbp)	%rcx	j1
%edx	-72(%rbp),		jmp	%rdx	.L16
%edx	subl		.L19	%rcx	leaq
%edx	\$1, %eax		movl	%rdx	-64(%rbp), %rax
%edx	cltq		-92(%rbp), %eax	%rcx	movq
%edx	movslq		movl	%rdx	%rax, %rsi
%edx	%edx, %rdx		-96(%rbp), %edx	%rcx	movl
%edx	\$3000,		movslq	%rdx	\$0, %edi
%edx	addq		%edx, %rdx	%rcx	call
%edx	%rdx, %rax		imulq	%rdx	clock_gettime@PLT
%edx	leaq		\$3000, %rdx,	%rdx	movl
%edx	0(, %rax, 4),		addq	%rdx	-108(%rbp),
%edx	leaq		%rdx, %rax	%rdx	leal
%edx	a(%rip), %rax		leaq	%rdx	3(%rax), %edx
%edx	(%rdx, %rax),		0(, %rax, 4),	%rdx	testl
%edx	movl		leaq	%rdx	%eax, %eax
%edx	-88(%rbp), %eax		a(%rip), %rax	%rdx	cmovs
%edx	cltq		movl	%rdx	%edx, %eax
%edx	movl		(%rdx, %rax),	%rdx	sarl
%edx	-96(%rbp), %ecx		movl	%rdx	\$2, %eax
%edx	%ecx, %rcx		-88(%rbp), %eax	%rdx	movl
%edx	imulq		cltq	%rdx	%eax, -88(%rbp)
%edx	\$3000, %rcx,		movl	%rdx	movl
%edx	addq		-104(%rbp),	%rdx	\$0, -104(%rbp)
%edx	%rcx, %rax		leaq	%rdx	jmp
%edx	leaq		-96(%rbp), %ecx	%rdx	.L17
%edx	0(, %rax, 4),		movslq	%rdx	movl
%edx	leaq		%ecx, %rcx	%rdx	\$0, -100(%rbp)
%edx	b(%rip), %rax		imulq	%rdx	jmp
%edx	(%rcx, %rax),		\$3000, %rcx,	%rdx	.L18
%edx	movl		addq	%rdx	movl
%edx	-88(%rbp), %eax		%rcx, %rax	%rdx	\$0, -96(%rbp)
%edx	cltq		leaq	%rdx	movl
%edx	movl		0(, %rax, 4),	%rdx	\$0, -92(%rbp)
%edx	-92(%rbp), %esi		leaq	%rdx	jmp
%edx			b(%rip), %rax	%rdx	.L19
%edx			(%rcx, %rax),	%rdx	movl
%edx			movl	%rdx	-100(%rbp),
%edx			-88(%rbp), %eax	%rdx	cltq
%edx			cltq	%rdx	movl
%edx			movl	%rdx	-104(%rbp),
%edx			-92(%rbp), %esi	%rdx	movslq

%rax	movl \$0, %eax call printf@PLT leaq -48(%rbp),	%rsi	movslq %esi, %rsi imulq \$3000, %rsi,	%rdx	%edx, %rdx imulq \$3000, %rdx,
	movq %rax, %rsi movl \$0, %edi call	%rsi	addq %rsi, %rax leaq 0(, %rax, 4),	%rdx	addq %rdx, %rax leaq 0(, %rax, 4),
clock_gettime@PLT	movq %rax, %rsi movl \$0, %edi call	%eax	leaq c(%rip), %rax movl (%rsi, %rax),	%edx	leaq a(%rip), %rax movl (%rdx, %rax),
68(%rbp)	movl \$0, -	%ecx	imull %ecx, %eax leal (%rdx, %rax),	%ecx	movl -96(%rbp), %eax cltq movl -104(%rbp),
.L18:	jmp .L13		movl -92(%rbp), %eax cltq movl -96(%rbp), %edx	%rcx	movslq %ecx, %rcx imulq \$3000, %rcx,
64(%rbp)	movl \$0, -	%rdx	movslq %edx, %rdx imulq \$3000, %rdx,	%rcx	addq %rcx, %rax leaq 0(, %rax, 4),
.L17:	jmp .L14	%rdx	addq %rdx, %rax leaq 0(, %rax, 4),	%ecx	leaq b(%rip), %rax movl (%rcx, %rax),
60(%rbp)	movl \$0, -	%rax)	leaq a(%rip), %rax movl %ecx, (%rdx,	%esi	movl -96(%rbp), %eax cltq movl -100(%rbp),
.L16:	jmp .L15	.L19:	addl \$1, -88(%rbp)	%rsi	movslq %esi, %rsi imulq \$3000, %rsi,
%eax	movl -64(%rbp),	%eax	movl -88(%rbp), %eax cmpl -100(%rbp),	%rsi	addq %rsi, %rax leaq 0(, %rax, 4),
%edx	cltq movl -68(%rbp),	.L18:	jl .L20 addl \$1, -92(%rbp)	%rsi	leaq c(%rip), %rax movl (%rsi, %rax),
%rdx, %rdx	movslq %edx, %rdx imulq \$3000,	%eax	movl -92(%rbp), %eax cmpl -100(%rbp),	%ecx	imull %ecx, %eax leal (%rdx, %rax),
%rdx	addq %rdx, %rax leaq 0(, %rax, 4),	.L17:	jl .L21 addl \$1, -96(%rbp)	%eax	movl -100(%rbp),
%rax	leaq a(%rip),	%eax	movl -96(%rbp), %eax cmpl -100(%rbp),	%edx	cltq movl -104(%rbp),
%rax), %edx	movl (%rdx,		jl .L22 leaq -48(%rbp), %rax movq %rax, %rsi movl \$0, %edi call	%rdx	movslq %edx, %rdx imulq \$3000, %rdx,
%eax	movl -60(%rbp),	clock_gettime@PLT	movq -48(%rbp), %rdx movq -64(%rbp), %rax subq %rax, %rdx movq %rdx, %rax cvtsi2sdq %rax, %xmm1 movq -40(%rbp), %rdx movq -56(%rbp), %rax subq %rax, %rdx movq %rdx, %rax cvtsi2sdq %rax, %xmm0 movsd .LC2(%rip),	%rdx	addq %rdx, %rax leaq 0(, %rax, 4),
%ecx	cltq movl -68(%rbp),		divsd %xmm2, %xmm0 addsd %xmm1, %xmm0 movsd %xmm0, -	%rdx	leaq a(%rip), %rax movl (%rdx, %rax),
%rcx, %rcx	movslq %ecx, %rcx imulq \$3000,	%xmm2		%edx	movl -100(%rbp),
%rcx	addq %rcx, %rax leaq 0(, %rax, 4),	80(%rbp)		%rdx	cltq movl -104(%rbp),
%rax	leaq b(%rip),			%rdx	movslq %edx, %rdx imulq \$3000, %rdx,
%rax), %ecx	movl (%rcx,			%rdx	addq %rdx, %rax leaq 0(, %rax, 4),
%eax	movl -64(%rbp),			%rdx	leaq a(%rip), %rax movl (%rdx, %rax),
%esi	cltq movl -60(%rbp),			%ecx	movl -96(%rbp), %eax addl \$1, %eax cltq movl -104(%rbp),
	movslq %esi, %rsi imulq		movq -48(%rbp), %rdx movq -32(%rbp), %rax		

%rsi, %rsi	\$3000,		subq %rax, %rdx		movslq %ecx, %rcx
	addq %rsi, %rax		movq %rdx, %rax	%rcx	imulq \$3000, %rcx,
%rsi	leaq 0(,%rax,4),		cvtsi2sdq %rax, %xmm1		addq %rcx, %rax
%rax	leaq c(%rip),		movq -40(%rbp), %rdx	%rcx	leaq 0(,%rax,4),
%rax), %eax	movl (%rsi,		subq %rax, %rdx		leaq b(%rip), %rax
	imull %ecx, %eax	%xmm2	movq %rdx, %rax	%ecx	movl (%rcx,%rax),
%rax), %ecx	leal (%rdx,		cvtsi2sdq %rax, %xmm0		movl -96(%rbp), %eax
%eax	movl -64(%rbp),	72(%rbp)	movsd %xmm0, -		addl \$1, %eax
%edx	cltq movl -68(%rbp),	%eax	divsd %xmm2, %xmm0	%esi	cltq movl -100(%rbp),
	movslq %edx, %rdx	%eax	addsd %xmm1, %xmm0		movslq %esi, %rsi
%rdx, %rdx	imulq \$3000,		movl -100(%rbp),	%rsi	addq %rsi, %rax
%rdx	addq %rdx, %rax	%rdx	leal -1(%rax), %edx	%rsi	leaq 0(,%rax,4),
%rdx	leaq a(%rip),	%rdx	movl -100(%rbp),	%eax	leaq c(%rip), %rax
(%rdx,%rax)	movl %ecx,	%edx	subl \$1, %eax	%ecx	movl (%rsi,%rax),
60(%rbp) .L15:	addl \$1, -		cltq movslq %edx, %rdx	%ecx	imull %ecx, %eax
%eax	movl -60(%rbp),	%rdi	imulq \$3000, %rdx,	%eax	leal (%rdx,%rax),
%eax	cmpl -72(%rbp),		addq %rdx, %rax	%edx	movl -100(%rbp),
	jl .L16	136(%rbp)	leaq a(%rip), %rax	%edx	cltq movl -104(%rbp),
64(%rbp) .L14:	addl \$1, -	%xmm0	movl (%rdx,%rax),	%rdx	movslq %edx, %rdx
%eax	movl -64(%rbp),	%rdi	movl a(%rip), %eax	%rdx	imulq \$3000, %rdx,
%eax	cmpl -72(%rbp),	%xmm0	movl %eax, %esi	%rdx	addq %rdx, %rax
	jl .L17	%xmm0	leaq .LC3(%rip),	%rdx	leaq 0(,%rax,4),
68(%rbp) .L13:	addl \$1, -	%xmm0	movl \$0, %eax	%rdx	leaq a(%rip), %rax
%eax	movl -68(%rbp),	136(%rbp)	call printf@PLT	%edx	movl (%rdx,%rax),
%eax	cmpl -72(%rbp),	%xmm0	movq -80(%rbp), %rax	%rdx	movl -96(%rbp), %eax
	jl .L18	%xmm0	movq %rax, -	%rdx	addl \$2, %eax
%rax	leaq -32(%rbp),	%rdi	movsd -136(%rbp),	%ecx	cltq movl -104(%rbp),
	movq %rax, %rsi		leaq .LC4(%rip),	%rcx	movslq %edx, %rdx
	movl \$0, %edi		movl \$1, %eax	%rcx	imulq \$3000, %rdx,
clock_gettime@PLT	call		call printf@PLT	%rcx	addq %rdx, %rax
	movq		movq -8(%rbp), %rdi	%rcx	leaq 0(,%rax,4),
			xorq %fs:40, %rdi	%rcx	leaq b(%rip), %rax
			je .L24	%ecx	movl (%rcx,%rax),
			call	%ecx	movl -96(%rbp), %eax
					addl

%rdx	-32(%rbp),	8	leave .cfi_def_cfa 7,	\$2, %eax
%rax	movq -48(%rbp),	.LFE6:	ret .cfi_endproc	cltq movl -100(%rbp),
	subq %rax, %rdx		.size main,.-main	movslq %esi, %rsi
	movq %rdx, %rax		.section .rodata	imulq \$3000, %rsi,
	cvtis2sdq %rax, %xmm1	.LC2:	.align 8	addq %rsi, %rax
%rdx	movq -24(%rbp),		.long 0	leaq 0(,%rax,4),
%rax	movq -40(%rbp),		.long 1104006501	leaq c(%rip), %rax
	subq %rax, %rdx	7.3.0-16ubuntu3)	ident "GCC: (Ubuntu	movl (%rsi,%rax),
	movq %rdx, %rax	stack,"",@progbits	.section .note.GNU-	imull %ecx, %eax
	movq %rdx, %rax			leal (%rdx,%rax),
	cvtis2sdq %rax, %xmm0			movl -100(%rbp),
%xmm2	movsd .LC3(%rip),			cltq movl -104(%rbp),
%xmm0	divsd %xmm2,			%edx
%xmm0	addsd %xmm1,			movslq %edx, %rdx
56(%rbp)	movsd %xmm0, -			imulq \$3000, %rdx,
%eax	movl -72(%rbp),			addq %rdx, %rax
%edx	leal -1(%rax),			leaq 0(,%rax,4),
%eax	movl -72(%rbp),			leaq a(%rip), %rax
	subl \$1, %eax			movl %ecx, (%rdx,
%rdx, %rdx	cltq movslq %edx, %rdx			movl -100(%rbp),
	imulq \$3000,			cltq movl -104(%rbp),
%rdx	addq %rdx, %rax			%edx
%rdx	leaq 0(,%rax,4),			movslq %edx, %rdx
%rax	leaq a(%rip),			imulq \$3000, %rdx,
%rax), %edx	movl (%rdx,			addq %rdx, %rax
%eax	movl a(%rip),			leaq 0(,%rax,4),
%rdi	movl %eax, %esi			leaq a(%rip), %rax
	leaq .LC4(%rip),			movl (%rdx,%rax),
	movl \$0, %eax			movl -96(%rbp), %eax
%rax	call printf@PLT			addl \$3, %eax
104(%rbp)	movq %rax, -			cltq movl -104(%rbp),
%xmm0	movsd -104(%rbp),			%ecx
%rdi	leaq .LC5(%rip),			movslq %ecx, %rcx
	movl			imulq \$3000, %rcx,
				addq %rcx, %rax
				leaq 0(,%rax,4),
				leaq b(%rip), %rax
				movl (%rcx,%rax),
				movl -96(%rbp), %eax
				addl \$3, %eax
				cltq movl -100(%rbp),
				%esi
				movslq %esi, %rsi
				imulq \$3000, %rsi,
				addq %rsi, %rax
				leaq 0(,%rax,4),
				%rsi
				leaq c(%rip), %rax
				movl

<pre> \$1, %eax call printf@PLT movl \$0, %eax movq -8(%rbp), %rdi xorq %fs:40, %rdi je .L20 call __stack_chk_fail@PLT .L20: leave .cfi_def_cfi a 7, 8 ret .cfi_endpro c .LFE6: .size main, .- main .section .rodata .align 8 .LC3: .long 0 .long 1104006501 .ident "GCC: (Ubuntu 7.3.0-16ubuntu3) 7.3.0" .section .note.GNU- stack,"",@progbits </pre>		<pre> (%rsi,%rax), %eax imull %ecx, %eax leal (%rdx,%rax), %ecx movl -100(%rbp), %eax cltq movl -104(%rbp), %edx movslq %edx, %rdx imulq \$3000, %rdx, %rdx addq %rdx, %rax leaq 0(,%rax,4), %rdx leaq a(%rip), %rax movl %ecx, (%rdx, %rax) addl \$4, -96(%rbp) addl \$1, -92(%rbp) .L19: movl -92(%rbp), %eax cmpl -88(%rbp), %eax jl .L20 movl -88(%rbp), %eax sall \$2, %eax movl %eax, -96(%rbp) jmp .L21 .L22: movl -100(%rbp), %eax cltq movl -104(%rbp), %edx movslq %edx, %rdx imulq \$3000, %rdx, %rdx addq %rdx, %rax leaq 0(,%rax,4), %rdx leaq a(%rip), %rax movl (%rdx,%rax), %edx movl -96(%rbp), %eax cltq movl -104(%rbp), %ecx movslq %ecx, %rcx imulq \$3000, %rcx, %rcx addq %rcx, %rax leaq 0(,%rax,4), %rcx leaq b(%rip), %rax movl (%rcx,%rax), %ecx movl -96(%rbp), %eax cltq movl -100(%rbp), %esi movslq %esi, %rsi imulq \$3000, %rsi, %rsi addq %rsi, %rax leaq 0(,%rax,4), %rsi leaq c(%rip), %rax movl (%rsi,%rax), %eax </pre>
---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------	--	---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

		<pre> imull %ecx, %eax leal (%rdx,%rax), %ecx movl -100(%rbp), %eax cltq movl -104(%rbp), %edx movslq %edx, %rdx imulq \$3000, %rdx, %rdx addq %rdx, %rax leaq 0(,%rax,4), %rdx leaq a(%rip), %rax movl %ecx, (%rdx, %rax) addl \$1, -96(%rbp) .L21: movl -96(%rbp), %eax cmpl -108(%rbp), %eax jl .L22 addl \$1, -100(%rbp) .L18: movl -100(%rbp), %eax cmpl -108(%rbp), %eax jl .L23 addl \$1, -104(%rbp) .L17: movl -104(%rbp), %eax cmpl -108(%rbp), %eax jl .L24 leaq -48(%rbp), %rax movq %rax, %rsi movl \$0, %edi call clock_gettime@PLT movq -48(%rbp), %rdx movq -64(%rbp), %rax subq %rax, %rdx movq %rdx, %rax cvtsi2sdq %rax, %xmm1 movq -40(%rbp), %rdx movq -56(%rbp), %rax subq %rax, %rdx movq %rdx, %rax cvtsi2sdq %rax, %xmm0 movsd .LC2(%rip), %xmm2 divsd %xmm2, %xmm0 addsd %xmm1, %xmm0 movsd %xmm0, - 80(%rbp) movq -48(%rbp), %rdx movq -32(%rbp), %rax subq %rax, %rdx movq %rdx, %rax cvtsi2sdq %rax, %xmm1 movq -40(%rbp), %rdx movq -24(%rbp), %rax subq </pre>
--	--	----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

		<pre> %rax, %rdx movq %rdx, %rax cvttsi2sdq %rax, %xmm0 movsd %xmm0, -108(%rbp), %ymm2 divsd %ymm2, %xmm0 addsd %xmm1, %xmm0 movsd %xmm0, - 72(%rbp) movl -108(%rbp), %eax leal -1(%rax), %edx movl -108(%rbp), %eax subl \$1, %eax cvttsi2sdq %eax, %rdx movslq %edx, %rdx imulq \$3000, %rdx, %rdx addq %rdx, %rax leaq 0(, %rax, 4), %rdx leaq a(%rip), %rax movl (%rdx, %rax), %edx movl a(%rip), %eax movl %eax, %esi leaq .LC3(%rip), %rdi movl \$0, %eax call printf@PLT movq -80(%rbp), %rax movq %rax, - 136(%rbp) movsd -136(%rbp), %ymm0 leaq .LC4(%rip), %rdi movl \$1, %eax call printf@PLT movsd -72(%rbp), %ymm0 subsd -80(%rbp), %ymm0 movq -72(%rbp), %rax movapd %ymm0, %xmm1 movq %rax, - 136(%rbp) movsd -136(%rbp), %ymm0 leaq .LC5(%rip), %rdi movl \$2, %eax call printf@PLT movl \$0, %eax movq -8(%rbp), %rdi xorq %fs:40, %rdi je .L26 call __stack_chk_fail@PLT .L26: leave .cfi_def_cfa 7, 8 ret .LFE6: .cfi_endproc .size main, .-main .section .rodata .align 8 </pre>
--	--	------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

		<pre> .LC2: .long 0 .long 1104006501 .ident "GCC: (Ubuntu 7.3.0-16ubuntu3) 7.3.0" .section .note.GNU- stack,"",@progbits </pre>
--	--	---------------------------------------------------------------------------------------------------------------------------------------------------------------------

B) CÓDIGO FIGURA 1:**CAPTURA CÓDIGO FUENTE:** figura1-original.c

```

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

#define MAX 40000

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

int main()
{
    int ii, i, X1, X2;
    int R[MAX];
    struct timespec cgt1,cgt2; double ncgt;

    clock_gettime(CLOCK_REALTIME,&cgt1);
    for(ii = 1; ii <= MAX;ii++){
        X1 = 0; X2 = 0;
        for(i = 0; i < 5000; i++)    X1+=2*s[i].a+ii;
        for(i = 0; i < 5000; i++)    X2+=3*s[i].b-ii;

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

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

    printf("R[0] = %i, R[MAX-1] = %i\n", R[0], R[MAX-1]);
    printf("\nTiempo (seg.) = %11.9f\n", ncgt);

    return 0;
}

```

1.1. MODIFICACIONES REALIZADAS (al menos dos modificaciones):

Modificación a) –explicación–: En la primera optimización, comprimo los dos bucles for que hay anidados en uno solo para ahorrar accesos a memoria. Otros cambios menores son ++ii e ++i en lugar de ii++ e i++ y cambiar el if por el operador ternario para ahorrar instrucciones.

Modificación b) –explicación–: En esta optimización realizo un desenrollado de bucle para 4 elementos.

1.1. CÓDIGOS FUENTE MODIFICACIONES**a) Captura figura1-modificado_a.c**

```

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

#define MAX 40000

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

int main()
{
    int ii, i, X1, X2;
    int R[MAX];
    struct timespec cgt1,cgt2; double ncgt;

```

```

clock_gettime(CLOCK_REALTIME,&cgt1);
for(ii = 1; ii <= MAX;++ii){
    X1 = 0; X2 = 0;
    for(i = 0; i < 5000; ++i){
        X1+=2*s[i].a+ii;
        X2+=3*s[i].b-ii;
    }
    /*if(X1<X2)?R[ii] = X1;
    else R[ii] = X2;*/
    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("R[0] = %i, R[MAX-1] = %i\n", R[0], R[MAX-1]);
printf("\nTiempo (seg.) = %11.9f\n", ncgt);

return 0;
}

```

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

```

~/Escritorio/segundo/ac/practicas/practica5 4:47:22
$ ./figural-original
R[0] = 0, R[MAX-1] = -199995000
Tiempo (seg.) = 0.259570052
~/Escritorio/segundo/ac/practicas/practica5 4:47:25
$ ./figural-modificado_a
R[0] = 0, R[MAX-1] = -199995000
Tiempo (seg.) = 0.194590940

```

b) Captura figura1-modificado_b.c

```

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

#define MAX 40000

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

int main()
{
    int ii, i, X1, X2;
    int R[MAX];
    struct timespec cgt1,cgt2; double ncgt;

    clock_gettime(CLOCK_REALTIME,&cgt1);
    for(ii = 1; ii <= MAX;++ii){
        X1 = 0; X2 = 0;
        for(i = 0; i < 5000; i=i+4){
            X1+=2*s[i].a+ii;
            X2+=3*s[i].b-ii;

            X1+=2*s[i+1].a+ii;
            X2+=3*s[i+1].b-ii;

            X1+=2*s[i+2].a+ii;

```

```

        X2+=3*s[i+2].b-ii;
        X1+=2*s[i+3].a+ii;
        X2+=3*s[i+3].b-ii;
    }
    /*if(X1<X2)?R[ii] = X1;
    else      R[ii] = X2;*/
    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("R[0] = %i, R[MAX-1] = %i\n", R[0], R[MAX-1]);
printf("\nTiempo (seg.) = %11.9f\n", ncgt);

return 0;
}

```

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

```

~/Escritorio/segundo/ac/practicas/practica5 4:48:08
$ ./figural-original
R[0] = 0, R[MAX-1] = -199995000
Tiempo (seg.) = 0.260318335
~/Escritorio/segundo/ac/practicas/practica5 4:48:13
$ ./figural-modificado b
R[0] = 0, R[MAX-1] = -199995000
Tiempo (seg.) = 0.151513611

```

1.1. TIEMPOS:

Modificación	-O0	-O1	-O2	-Os
Sin modificar	1.096445224	0.347535642	0.358827921	0.346570367
Modificación a)	0.796539088	0.262023938	0.275537938	0.248625560
Modificación b)	0.656120900	0.208614030	0.222736926	0.235394405

1.1. COMENTARIOS SOBRE LOS RESULTADOS:

Si comparamos el tiempo de ejecución de la versión sin modificar y la modificación A, vemos que hay una diferencia bastante significativa, esto se consigue simplemente fusionando dos bucles y así nos ahorramos recorrer el mismo bucle dos veces.

Por otro lado, no hay mucha diferencia entre la modificación A y la B, pero con el desenrollado de bucle favorecemos la generación de instrucciones independientes haciendo más eficiente la ejecución del programa.

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

Figura 1 original	Figura 1 modificado a	Figura 1 modificado b
<code>.file "figura1-</code>	<code>.file "figura1-</code>	<code>.file "figura1-</code>

original.c"	<pre> .text .comm s,40000,32 .section .rodata .LC1: .string "R[0] = %i, R[MAX-1] = %i\n" .LC2: .string "\nTiempo (seg.) = %11.9f\n" .text .globl main .type main, @function main: .LFB0: .cfi_startproc pushq %rbp .cfi_def_cfa_of fset 16 .cfi_offset 6, -16 movq %rsp, %rbp .cfi_def_cfa_re gister 6 subq \$160096, %rsp movq %fs:40, %rax movq %rax, -8(%rbp) xorl %eax, %eax leaq -160048(%rbp), %rax movq %rax, %rsi movl \$0, %edi call clock_gettime@PLT movl \$1, - 160072(%rbp) jmp .L2 .L9: movl \$0, - 160064(%rbp) movl \$0, - 160060(%rbp) movl \$0, - 160068(%rbp) jmp .L3 .L4: movl -160068(%rbp), %eax cltq leaq 0(,%rax,8), %rdx leaq s(%rip), %rax movl (%rdx,%rax), %eax leal (%rax,%rax), %edx movl -160072(%rbp), %eax addl %edx, %eax addl %eax, - 160064(%rbp) addl \$1, - 160068(%rbp) .L3: cmpl \$4999, - 160068(%rbp) jle .L4 movl \$0, - 160068(%rbp) jmp .L5 .L6: movl -160068(%rbp), %eax cltq leaq 0(,%rax,8), %rdx </pre>	modificado_a.c"	<pre> .text .comm s,40000,32 .section .rodata .LC1: .string "R[0] = %i, R[MAX-1] = %i\n" .LC2: .string "\nTiempo (seg.) = %11.9f\n" .text .globl main .type main, @function main: .LFB0: .cfi_startproc pushq %rbp .cfi_def_cfa_of fset 16 .cfi_offset 6, -16 movq %rsp, %rbp .cfi_def_cfa_re gister 6 subq \$160096, %rsp movq %fs:40, %rax movq %rax, -8(%rbp) xorl %eax, %eax leaq -160048(%rbp), %rax movq %rax, %rsi movl \$0, %edi call clock_gettime@PLT movl \$1, - 160072(%rbp) jmp .L2 .L5: movl \$0, - 160064(%rbp) movl \$0, - 160060(%rbp) movl \$0, - 160068(%rbp) jmp .L3 .L4: movl -160068(%rbp), %eax cltq leaq 0(,%rax,8), %rdx leaq s(%rip), %rax movl (%rdx,%rax), %eax leal (%rax,%rax), %edx movl -160072(%rbp), %eax addl %edx, %eax addl %eax, - 160064(%rbp) movl -160068(%rbp), %eax cltq leaq 0(,%rax,8), %rdx leaq 4+s(%rip), %rax movl (%rdx,%rax), %edx movl %edx, %eax addl %eax, %eax addl %edx, %eax subl -160072(%rbp), %eax addl </pre>	modificado_b.c"	<pre> .text .comm s,40000,32 .section .rodata .LC1: .string "R[0] = %i, R[MAX-1] = %i\n" .LC2: .string "\nTiempo (seg.) = %11.9f\n" .text .globl main .type main, @function main: .LFB0: .cfi_startproc pushq %rbp .cfi_def_cfa_of fset 16 .cfi_offset 6, -16 movq %rsp, %rbp .cfi_def_cfa_re gister 6 subq \$160096, %rsp movq %fs:40, %rax movq %rax, -8(%rbp) xorl %eax, %eax leaq -160048(%rbp), %rax movq %rax, %rsi movl \$0, %edi call clock_gettime@PLT movl \$1, - 160072(%rbp) jmp .L2 .L5: movl \$0, - 160064(%rbp) movl \$0, - 160060(%rbp) movl \$0, - 160068(%rbp) jmp .L3 .L4: movl -160068(%rbp), %eax cltq leaq 0(,%rax,8), %rdx leaq s(%rip), %rax movl (%rdx,%rax), %eax leal (%rax,%rax), %edx movl -160072(%rbp), %eax addl %edx, %eax addl %eax, - 160064(%rbp) movl -160068(%rbp), %eax cltq leaq 0(,%rax,8), %rdx leaq 4+s(%rip), %rax movl (%rdx,%rax), %edx movl %edx, %eax addl %eax, %eax addl %edx, %eax subl -160072(%rbp), %eax addl </pre>
-------------	---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------	-----------------	---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------	-----------------	---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

%edx	leaq 4+s(%rip), %rax	160060(%rbp)	%eax, -	160060(%rbp)	%eax, -
	movl (%rdx,%rax),	addl \$1, -		movl -160068(%rbp),	
%eax	movl %edx, %eax	160068(%rbp)	.L3:	addl \$1,%eax	
	addl %eax, %eax	160068(%rbp)	cmp1 \$4999, -	cltq leaq 0(,%rax,8),	
160060(%rbp)	addl %edx, %eax	%eax	jle .L4	leaq s(%rip), %rax	
	subl -160072(%rbp),	160060(%rbp)	movl -160064(%rbp),	movl (%rdx,%rax),	
160068(%rbp)	addl %eax, -	%eax	cmp1 %eax, -	leal (%rax,%rax),	
.L5:	addl \$1, -	160060(%rbp)	cmovle -160060(%rbp),	%edx movl -160072(%rbp),	
160068(%rbp)	cmp1 \$4999, -	%eax	movl %eax, %edx	addl %edx, %eax	
%eax	jle .L6	160016(%rbp,%rax,4)	movl -160072(%rbp),	addl %eax, -	
	movl -160064(%rbp),	160072(%rbp)	cltq movl %edx, -	movl -160068(%rbp),	
%eax	cmp1 -160060(%rbp),	.L2:	addl \$1, -	addl \$1,%eax	
	jge .L7	160072(%rbp)	cmp1 \$40000, -	cltq leaq 0(,%rax,8),	
%eax	movl -160072(%rbp),	%rax	jle .L5	leaq 4+s(%rip), %rax	
	cltq movl -160064(%rbp),	clock_gettime@PLT	leaq -160032(%rbp),	movl (%rdx,%rax),	
%edx	movl %edx, -	movq %rax, %rsi	movl \$0, %edi	movl %edx, %eax	
160016(%rbp,%rax,4)	jmp .L8	call		addl %eax, %eax	
.L7:	movl -160072(%rbp),	movq -160032(%rbp),		addl %edx, %eax	
%eax	cltq movl -160060(%rbp),	movq -160048(%rbp),		subl -160072(%rbp),	
%edx	movl %edx, -	subq %rax, %rdx		addl %eax, -	
160016(%rbp,%rax,4)	addl \$1, -	movq %rdx, %rax		movl -160068(%rbp),	
160072(%rbp)	.L8:	cvtsi2sdq %rax, %xmm1		addl \$2,%eax	
.L2:	cmp1 \$40000, -	movq -160024(%rbp),		cltq leaq 0(,%rax,8),	
160072(%rbp)	jle .L9	movq -160040(%rbp),		rdx leaq s(%rip), %rax	
%rax	leaq -160032(%rbp),	subq %rax, %rdx		movl (%rdx,%rax),	
	movq %rax, %rsi	movq %rdx, %rax		leal (%rax,%rax),	
clock_gettime@PLT	movl \$0, %edi	cvtsi2sdq %rax, %xmm0		%edx movl -160072(%rbp),	
	call	movsd .LC0(%rip),		addl %edx, %eax	
%rdx	movq -160032(%rbp),	divsd %xmm2, %xmm0		addl %eax, -	
%rax	movq -160048(%rbp),	addsd %xmm1, %xmm0		movl -160068(%rbp),	
	subq %rax, %rdx	movsd %xmm0, -		addl \$2,%eax	
%rdx	movq %rdx, %rax	movl -20(%rbp), %edx		cltq leaq 0(,%rax,8),	
	cvtsi2sdq %rax, %xmm1	movl -160016(%rbp),		rdx leaq 4+s(%rip), %rax	
%rdx	movq -160024(%rbp),	movl %eax, %esi		movl (%rdx,%rax),	
	movq -160040(%rbp),	leaq .LC1(%rip),		%edx movl %edx, %eax	
%rax	subq %rax, %rdx	call printf@PLT		addl %eax, %eax	
	movq %rdx, %rax	movq -160056(%rbp),		addl %edx, %eax	
%xmm2	cvtsi2sdq %rax, %xmm0	movq %rax, -		subl -160072(%rbp),	
	movsd .LC0(%rip),	movsd -160088(%rbp),		addl %eax, -	
160056(%rbp)	divsd %xmm2, %xmm0	leaq .LC2(%rip),		movl -160068(%rbp),	
	addsd %xmm1, %xmm0	movl \$1,%eax		cltq leaq 0(,%rax,8),	
	movsd %xmm0, -	call printf@PLT			
	movl -20(%rbp), %edx	movl			

%eax	movl -160016(%rbp),	\$0, %eax movq -8(%rbp), %rcx xorq %fs:40, %rcx je .L7 call	leaq s(%rip), %rax movl (%rdx,%rax),
%rdi	leaq .LC1(%rip),		leal (%rax,%rax),
	movl \$0, %eax call printf@PLT movq -160056(%rbp),	__stack_chk_fail@PLT .L7: leave .cfi_def_cfa 7, ret .cfi_endproc	%edx movl -160072(%rbp),
%rax	movq %rax, -	.LFE0: .size main, .-main .section .rodata .align 8	%eax addl %edx, %eax addl %eax, -
160088(%rbp)	movsd -160088(%rbp),		movl -160068(%rbp),
%xmm0	leaq .LC2(%rip),	.LC0: .long 0 .long 1104006501 .ident "GCC: (Ubuntu 7.3.0-16ubuntu3) 7.3.0" .section .note.GNU-	%eax addl \$3, %eax cltq leaq 0(,%rax,8),
%rdi	movl \$1, %eax call printf@PLT movl \$0, %eax movq -8(%rbp), %rcx xorq %fs:40, %rcx je .L11 call	stack,"",@progbits	%rdx leaq 4+s(%rip), %rax movl (%rdx,%rax),
	__stack_chk_fail@PLT .L11: leave .cfi_def_cfa 7, ret .cfi_endproc		%edx movl %edx, %eax addl %eax, %eax addl %edx, %eax subl -160072(%rbp),
8			%eax addl %eax, -
.LFE0:	.size main, .-main .section .rodata .align 8		160060(%rbp) addl \$4, -
.LC0:	.long 0 .long 1104006501 .ident "GCC: (Ubuntu 7.3.0-16ubuntu3) 7.3.0" .section .note.GNU-		160068(%rbp) .L3: cmpl \$4999, -
7.3.0-16ubuntu3)			160068(%rbp) jle .L4 movl -160064(%rbp),
stack,"",@progbits			%eax cmpl %eax, -
			160060(%rbp) cmovle -160060(%rbp),
			%eax movl %eax, %edx movl -160072(%rbp),
			%eax cltq movl %edx, -
			160016(%rbp,%rax,4) addl \$1, -
			160072(%rbp) .L2: cmpl \$40000, -
			160072(%rbp) jle .L5 leaq -160032(%rbp),
			%rax movq %rax, %rsi movl \$0, %edi call
			clock_gettime@PLT movq -160032(%rbp),
			%rdx movq -160048(%rbp),
			%rax subq %rax, %rdx movq %rdx, %rax cvtsi2sdq %rax, %xmm1 movq -160024(%rbp),
			%rdx movq -160040(%rbp),
			%rax subq %rax, %rdx movq %rdx, %rax cvtsi2sdq %rax, %xmm0 movsd

		<pre> %xmm2 .LC0(%rip), divsd %xmm2, %xmm0 addsd %xmm1, %xmm0 movsd %xmm0, - 160056(%rbp) movl -20(%rbp), %edx movl -160016(%rbp), %eax movl %eax, %esi leaq .LC1(%rip), %rdi movl \$0, %eax call printf@PLT movq -160056(%rbp), %rax movq %rax, - 160088(%rbp) movsd -160088(%rbp), %xmm0 leaq .LC2(%rip), %rdi movl \$1, %eax call printf@PLT movl \$0, %eax movq -8(%rbp), %rcx xorq %fs:40, %rcx je .L7 call __stack_chk_fail@PLT .L7: leave .cfi_def_cfa 7, 8 ret .cfi_endproc .LFE0: .size main, .-main .section .rodata .align 8 .LC0: .long 0 .long 1104006501 .ident "GCC: (Ubuntu 7.3.0-16ubuntu3) .section .note.GNU- stack,"",@progbits </pre>
--	--	--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

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<=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 acarrearán. Incorpore los códigos al cuaderno de prácticas y destaque las diferencias entre ellos.

2.2. (Ejercicio EXTRA) Para la mejor de las opciones, obtenga los tiempos de ejecución con distintos valores de N y determine para su sistema los valores de Rmax (valor máximo del número de operaciones en coma flotante por unidad de tiempo), Nmax (valor de N para el que se consigue Rmax), y N1/2 (valor de N para el que se obtiene Rmax/2). Estime el

valor de la velocidad pico (Rpico) del procesador (consulte en [4] el número de ciclos por instrucción punto flotante para la familia y modelo de procesador que está utilizando) y compárela con el valor obtenido para Rmax. -Consulte la Lección 3 del Tema 1.

CAPTURA CÓDIGO FUENTE: daxpy.c

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

int main(int argc, char** argv)
{
    int i, N, a;
    int *x, *y;
    struct timespec cgt1, cgt2; double ncgt;

    if(argc<3){ fprintf(stderr, "Error en argumentos> ./daxpy [tamaño
vector] [constante]\n"); exit(-1);}

    N = atoi(argv[1]);
    y = (int*) malloc(N*sizeof(int));
    x = (int*) malloc(N*sizeof(int));

    a = atoi(argv[2]);
    for(i=0; i<N;i++){ x[i] = i; y[i] = i*2;}
    clock_gettime(CLOCK_REALTIME, &cgt1);

    for(i = 0; i < N; i++){
        y[i] = a*x[i]+y[i];
    }

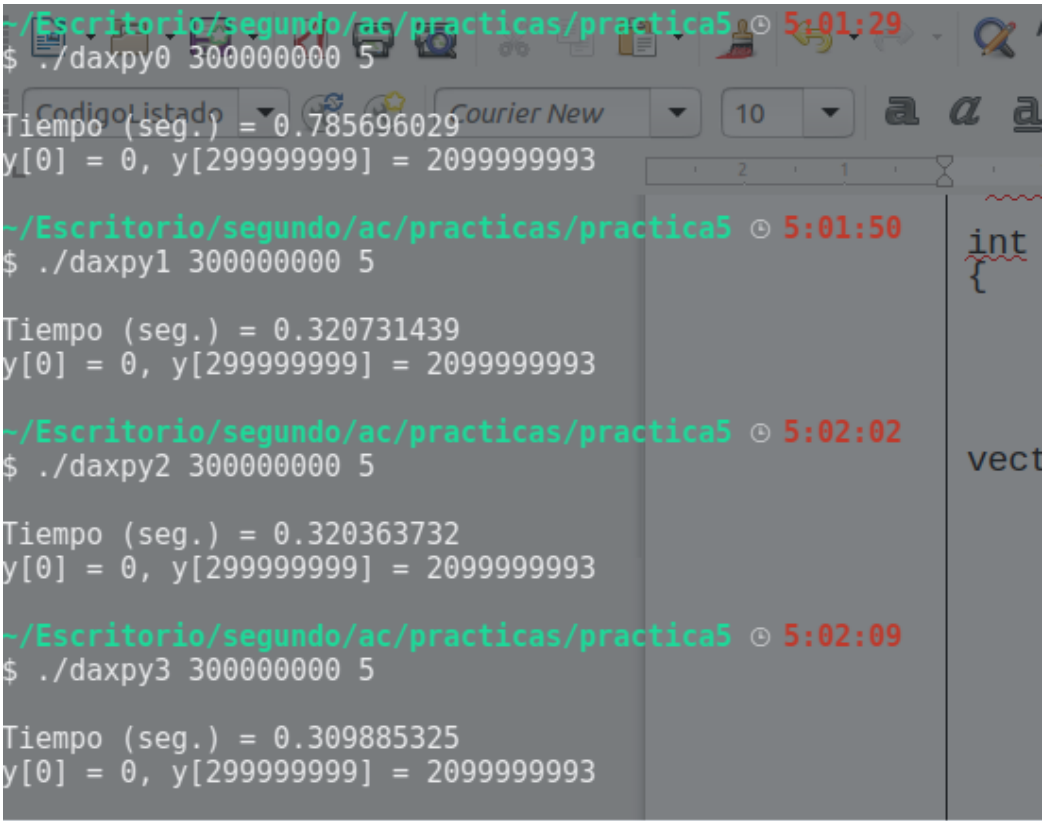
    clock_gettime(CLOCK_REALTIME, &cgt2);
    ncgt=(double) (cgt2.tv_sec-cgt1.tv_sec)+( double) ((cgt2.tv_nsec-
cgt1.tv_nsec)/(1.e+9));

    printf("\nTiempo (seg.) = %11.9f\n", ncgt);
    printf("y[0] = %i, y[%d] = %i\n", y[0], N-1, y[N-1]);

    return 0;
}
```

	-O0	-Os	-O2	-O3
Tiempos ejec.	0.785696 029	0.320731 439	0.320363 732	0.309885 325

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



COMENTARIOS QUE EXPLIQUEN LAS DIFERENCIAS EN ENSAMBLADOR:

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

daxpy00.s	daxpy0s.s	daxpy02.s	daxpy03.s