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 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;
   }
   …
}
```

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 tamanio 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 ejmplo:
     * 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 trasponer 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 desenrrollado de bucle de 4 en 4 saltos.

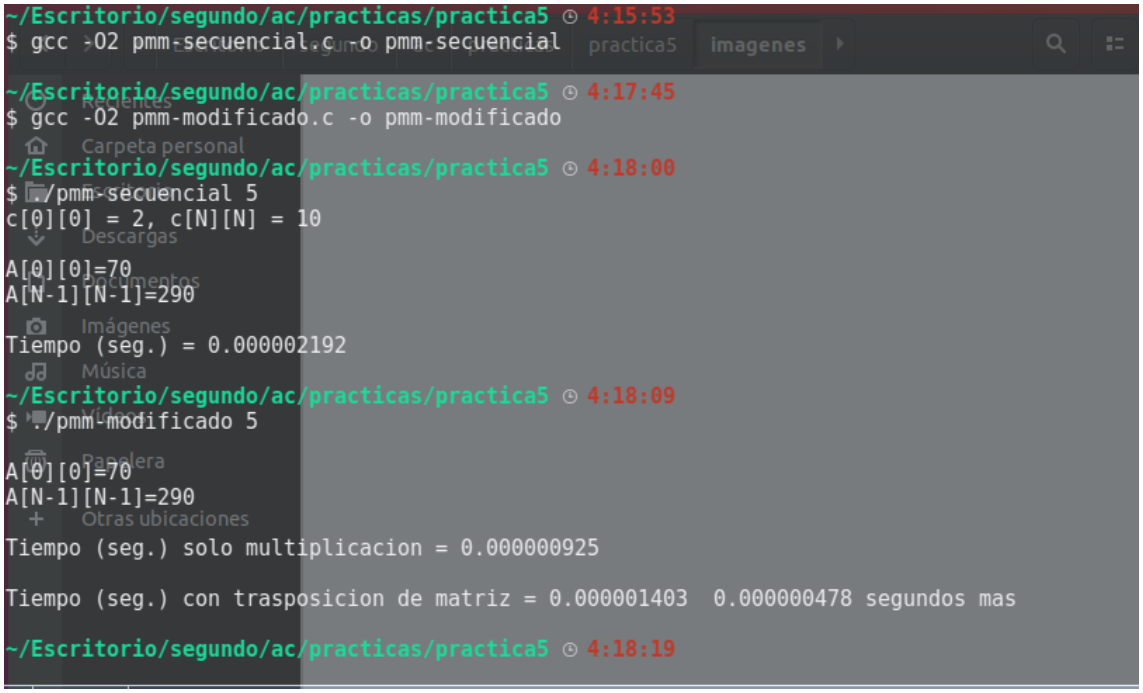
1.1. CÓDIGOS FUENTE MODIFICACIONES

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 tamanio 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 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):

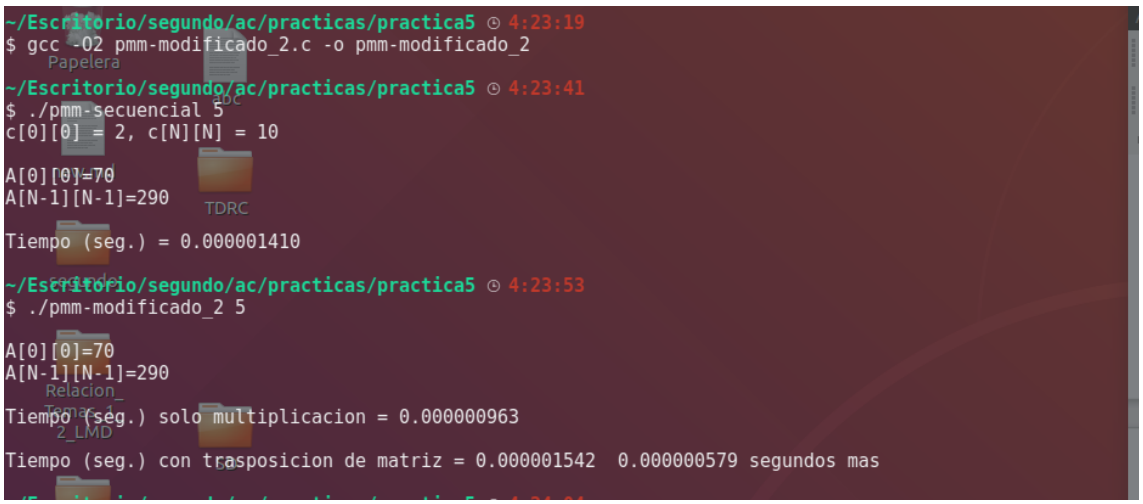


b) Captura de `pmm-secuencial-modificado_b.c`

```
#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 tamanio 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):



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 desenrrollado 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

```
	.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
imprimirMatriz:
.LFB5:
	.cfi_startproc
	pushq	%rbp
	.cfi_def_cfa_offset 16
	.cfi_offset 6, -16
	movq	%rsp, %rbp
	.cfi_def_cfa_register 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
```

pmm-secuencial-modificado_b.s

```
	.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_offset 16
	.cfi_offset 6, -16
	movq	%rsp, %rbp
	.cfi_def_cfa_register 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
```

pmm-secuencial-modificado_c.s

```
	.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_offset 16
	.cfi_offset 6, -16
	movq	%rsp, %rbp
	.cfi_def_cfa_register 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
```

```
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    %rax, %rdx
        movl    -4(%rbp), %eax
        cltq
        movl    (%rdx, %rax,4), %eax
        movl    %eax, %esi
        leaq    .LC0(%rip), %rdi
        movl    $0, %eax
        call    printf@PLT
        addl    $1, -4(%rbp)
.L3:
        movl    -4(%rbp), %eax
        cmpl    -28(%rbp), %eax
        jl      .L4
        movl    $10, %edi
        call    putchar@PLT
        addl    $1, -8(%rbp)
.L2:
        movl    -8(%rbp), %eax
        cmpl    -28(%rbp), %eax
        jl      .L5
        movl    $10, %edi
        call    putchar@PLT
        nop
        leave
        .cfi_def_cfa 7, 8
        ret
        .cfi_endproc
.LFE5:
        .size   imprimirMatriz, .-imprimirMatriz
        .section        .rodata
        .align 8
```

```
        movl    %rax, %rdx
        movl    -4(%rbp), %eax
        cltq
        movl    (%rdx,%rax,4), %eax
        movl    %eax, %esi
        leaq    .LC0(%rip), %rdi
        movl    $0, %eax
        call    printf@PLT
        addl    $1, -4(%rbp)
.L3:
        movl    -4(%rbp), %eax
        cmpl    -28(%rbp), %eax
        jl      .L4
        movl    $10, %edi
        call    putchar@PLT
        addl    $1, -8(%rbp)
.L2:
        movl    -8(%rbp), %eax
        cmpl    -28(%rbp), %eax
        jl      .L5
        movl    $10, %edi
        call    putchar@PLT
        nop
        leave
        .cfi_def_cfa 7, 8
        ret
        .cfi_endproc
.LFE5:
        .size   imprimirMatriz, .-imprimirMatriz
        .section        .rodata
        .align 8
.LC1:
        .string "Falta el tamanio de la matriz\n"
.LC3:
        .string "\nA[0][0]=%d\nA[N-1][N-1]=%d\n"
        .align 8
.LC4:
        .string "\nTiempo (seg.) solo multiplicacion = %11.9f\n"
        .align 8
.LC5:
        .string "\nTiempo (seg.) con trasposicion de matriz = %11.9f\t%11.9f segundos mas\n"
        .text
        .globl  main
        .type   main, @function
main:
.LFB6:
        .cfi_startproc
        pushq   %rbp
        .cfi_def_cfa_offset 16
        .cfi_offset 6, -16
        movq    %rsp, %rbp
        .cfi_def_cfa_register 6
        subq    $144, %rsp
        movl    %edi, -116(%rbp)
        movq    %rsi, -128(%rbp)
        movq    %fs:40, %rax
        movq    %rax, -8(%rbp)
        xorl    %eax, %eax
        cmpl    $1, -116(%rbp)
        jg      .L7
        movq
```

```
        movl    %rax, %rdx
        movl    -4(%rbp), %eax
        cltq
        movl    (%rdx,%rax,4), %eax
        movl    %eax, %esi
        leaq    .LC0(%rip), %rdi
        movl    $0, %eax
        call    printf@PLT
        addl    $1, -4(%rbp)
.L3:
        movl    -4(%rbp), %eax
        cmpl    -28(%rbp), %eax
        jl      .L4
        movl    $10, %edi
        call    putchar@PLT
        addl    $1, -8(%rbp)
.L2:
        movl    -8(%rbp), %eax
        cmpl    -28(%rbp), %eax
        jl      .L5
        movl    $10, %edi
        call    putchar@PLT
        nop
        leave
        .cfi_def_cfa 7, 8
        ret
        .cfi_endproc
.LFE5:
        .size   imprimirMatriz, .-imprimirMatriz
        .section        .rodata
        .align 8
.LC1:
        .string "Falta el tamanio de la matriz\n"
.LC3:
        .string "\nA[0][0]=%d\nA[N-1][N-1]=%d\n"
        .align 8
.LC4:
        .string "\nTiempo (seg.) solo multiplicacion = %11.9f\n"
        .align 8
.LC5:
        .string "\nTiempo (seg.) con trasposicion de matriz = %11.9f\t%11.9f segundos mas\n"
        .text
        .globl  main
        .type   main, @function
main:
.LFB6:
        .cfi_startproc
        pushq   %rbp
        .cfi_def_cfa_offset 16
        .cfi_offset 6, -16
        movq    %rsp, %rbp
        .cfi_def_cfa_register 6
        subq    $144, %rsp
        movl    %edi, -116(%rbp)
        movq    %rsi, -128(%rbp)
        movq    %fs:40, %rax
        movq    %rax, -8(%rbp)
        xorl    %eax, %eax
        cmpl    $1, -116(%rbp)
        jg      .L7
        movq
```

```
.LC1:
        .string "Falta el tamanio 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_startproc
        pushq   %rbp
        .cfi_def_cfa_offset 16
        .cfi_offset 6, -16
        movq    %rsp, %rbp
        .cfi_def_cfa_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, -100(%rbp)
        jle     .L8
        movl    $3000, -100(%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
```

```
        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,%rax)
        movl    -104(%rbp), %eax
        leal    2(%rax), %edx
        movl    -100(%rbp), %eax
        leal
```

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

```
                -64(%rbp),
%eax
                leal
                (%rdx,
%rax), %ecx
                movl
                -64(%rbp),
%eax
                cltq
                movl
                -68(%rbp),
%edx
                movslq
                %edx, %rdx
                imulq
                $3000,
%rdx, %rdx
                addq
                %rdx, %rax
                leaq
                0(,%rax,4),
%rdx
                leaq
                c(%rip),
%rax
                movl
                %ecx,
(%rdx,%rax)
                addl
                $1, -
64(%rbp)
.L10:
                movl
                -64(%rbp),
%eax
                cmpl
                -72(%rbp),
%eax
                jl
                .L11
                addl
                $1, -
68(%rbp)
.L9:
                movl
                -68(%rbp),
%eax
                cmpl
                -72(%rbp),
%eax
                jl
                .L12
                movl
                -72(%rbp),
%eax
                leal
                -1(%rax),
%edx
                movl
                -72(%rbp),
%eax
                subl
                $1, %eax
                cltq
                movslq
                %edx, %rdx
                imulq
                $3000,
%rdx, %rdx
                addq
                %rdx, %rax
                leaq
                0(,%rax,4),
%rdx
                leaq
                c(%rip),
%rax
                movl
                (%rdx,
%rax), %edx
                movl
                c(%rip),
%eax
                movl
                %eax, %esi
                leaq
                .LC2(%rip),
%rdi
                %edx, (%rcx,
%rax)
                movl
                -96(%rbp), %eax
                cltq
                movl
                -92(%rbp), %edx
                movslq
                %edx, %rdx
                imulq
                $3000, %rdx,
%rdx
                addq
                %rdx, %rax
                leaq
                0(,%rax,4),
%rcx
                leaq
                c(%rip), %rax
                movl
                -84(%rbp), %edx
                movl
                %edx, (%rcx,
%rax)
                addl
                $1, -92(%rbp)
.L14:
                movl
                -92(%rbp), %eax
                cmpl
                -100(%rbp),
%eax
                jl
                .L15
                addl
                $1, -96(%rbp)
.L13:
                movl
                -96(%rbp), %eax
                cmpl
                -100(%rbp),
%eax
                jl
                .L16
                leaq
                -64(%rbp), %rax
                movq
                %rax, %rsi
                movl
                $0, %edi
                call
clock_gettime@PLT
                movl
                $0, -96(%rbp)
                jmp
                .L17
.L22:
                movl
                $0, -92(%rbp)
                jmp
                .L18
.L21:
                movl
                $0, -88(%rbp)
                jmp
                .L19
.L20:
                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
                (%rdx,%rax),
%edx
                movl
                -88(%rbp), %eax
                cltq
                movl
                -96(%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
                -88(%rbp), %eax
                cltq
                movl
                -92(%rbp), %esi
                %eax
                cltq
                movl
                -104(%rbp),
%ecx
                movslq
                %ecx, %rcx
                imulq
                $3000, %rcx,
%rcx
                addq
                %rcx, %rax
                leaq
                0(,%rax,4),
%rcx
                leaq
                c(%rip), %rax
                movl
                %edx, (%rcx,
%rax)
                movl
                -104(%rbp),
%eax
                cltq
                movl
                -100(%rbp),
%edx
                movslq
                %edx, %rdx
                imulq
                $3000, %rdx,
%rdx
                addq
                %rdx, %rax
                leaq
                0(,%rax,4),
%rcx
                leaq
                c(%rip), %rax
                movl
                -84(%rbp), %edx
                movl
                %edx, (%rcx,
%rax)
                addl
                $1, -100(%rbp)
.L14:
                movl
                -100(%rbp),
%eax
                cmpl
                -108(%rbp),
%eax
                jl
                .L15
                addl
                $1, -104(%rbp)
.L13:
                movl
                -104(%rbp),
%eax
                cmpl
                -108(%rbp),
%eax
                jl
                .L16
                leaq
                -64(%rbp), %rax
                movq
                %rax, %rsi
                movl
                $0, %edi
                call
clock_gettime@PLT
                movl
                -108(%rbp),
%eax
                leal
                3(%rax), %edx
                testl
                %eax, %eax
                cmovs
                %edx, %eax
                sarl
                $2, %eax
                movl
                %eax, -88(%rbp)
                movl
                $0, -104(%rbp)
                jmp
                .L17
.L24:
                movl
                $0, -100(%rbp)
                jmp
                .L18
.L23:
                movl
                $0, -96(%rbp)
                movl
                $0, -92(%rbp)
                jmp
                .L19
.L20:
                movl
                -100(%rbp),
%eax
                cltq
                movl
                -104(%rbp),
%edx
                movslq
```

```
                movl
                $0, %eax
                call
                printf@PLT
                leaq
                -48(%rbp),
%rax
                movq
                %rax, %rsi
                movl
                $0, %edi
                call
clock_gettime@PLT
                movl
                $0, -
68(%rbp)
                jmp
                .L13
.L18:
                movl
                $0, -
64(%rbp)
                jmp
                .L14
.L17:
                movl
                $0, -
60(%rbp)
                jmp
                .L15
.L16:
                movl
                -64(%rbp),
%eax
                cltq
                movl
                -68(%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
                -60(%rbp),
%eax
                cltq
                movl
                -68(%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
                -64(%rbp),
%eax
                cltq
                movl
                -60(%rbp),
%esi
                movslq
                %esi, %rsi
                imulq
                movslq
                %esi, %rsi
                imulq
                $3000, %rsi,
%rsi
                addq
                %rsi, %rax
                leaq
                0(,%rax,4),
%rsi
                leaq
                c(%rip), %rax
                movl
                (%rsi,%rax),
%eax
                imull
                %ecx, %eax
                leal
                (%rdx,%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
                a(%rip), %rax
                movl
                %ecx, (%rdx,
%rax)
                addl
                $1, -88(%rbp)
.L19:
                movl
                -88(%rbp), %eax
                cmpl
                -100(%rbp),
%eax
                jl
                .L20
                addl
                $1, -92(%rbp)
.L18:
                movl
                -92(%rbp), %eax
                cmpl
                -100(%rbp),
%eax
                jl
                .L21
                addl
                $1, -96(%rbp)
.L17:
                movl
                -96(%rbp), %eax
                cmpl
                -100(%rbp),
%eax
                jl
                .L22
                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
                %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
                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)
                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
                addl
                $1, %eax
                cltq
                movl
                -104(%rbp),
%ecx
```

```
                $3000,
%rsi, %rsi
                addq
                %rsi, %rax
                leaq
                0(,%rax,4),
%rsi
                leaq
                c(%rip),
%rax
                movl
                (%rsi,
%rax), %eax
                imull
                %ecx, %eax
                leal
                (%rdx,
%rax), %ecx
                movl
                -64(%rbp),
%eax
                cltq
                movl
                -68(%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, -
60(%rbp)
.L15:
                movl
                -60(%rbp),
%eax
                cmpl
                -72(%rbp),
%eax
                jl
                .L16
                addl
                $1, -
64(%rbp)
.L14:
                movl
                -64(%rbp),
%eax
                cmpl
                -72(%rbp),
%eax
                jl
                .L17
                addl
                $1, -
68(%rbp)
.L13:
                movl
                -68(%rbp),
%eax
                cmpl
                -72(%rbp),
%eax
                jl
                .L18
                leaq
                -32(%rbp),
%rax
                movq
                %rax, %rsi
                movl
                $0, %edi
                call
clock_gettime@PLT
                movq
                subq
                %rax, %rdx
                movq
                %rdx, %rax
                cvtsi2sdq
                %rax, %xmm1
                movq
                -40(%rbp), %rdx
                movq
                -24(%rbp), %rax
                subq
                %rax, %rdx
                movq
                %rdx, %rax
                cvtsi2sdq
                %rax, %xmm0
                movsd
                .LC2(%rip),
%xmm2
                divsd
                %xmm2, %xmm0
                addsd
                %xmm1, %xmm0
                movsd
                %xmm0, -
72(%rbp)
                movl
                -100(%rbp),
%eax
                leal
                -1(%rax), %edx
                movl
                -100(%rbp),
%eax
                subl
                $1, %eax
                cltq
                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),
%xmm0
                leaq
                .LC4(%rip),
%rdi
                movl
                $1, %eax
                call
                printf@PLT
                movsd
                -72(%rbp),
%xmm0
                subsd
                -80(%rbp),
%xmm0
                movq
                -72(%rbp), %rax
                movapd
                %xmm0, %xmm1
                movq
                %rax, -
136(%rbp)
                movsd
                -136(%rbp),
%xmm0
                leaq
                .LC5(%rip),
%rdi
                movl
                $2, %eax
                call
                printf@PLT
                movl
                $0, %eax
                movq
                -8(%rbp), %rdi
                xorq
                %fs:40, %rdi
                je
                .L24
                call
__stack_chk_fail@PLT
.L24:
                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
                addl
                $1, %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
                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)
                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
                addl
                $2, %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
                addl
```

```
                -32(%rbp),
%rdx
                movq
                -48(%rbp),
%rax
                subq
                %rax, %rdx
                movq
                %rdx, %rax
                cvtsi2sdq
                %rax, %xmm1
                movq
                -24(%rbp),
%rdx
                movq
                -40(%rbp),
%rax
                subq
                %rax, %rdx
                movq
                %rdx, %rax
                cvtsi2sdq
                %rax, %xmm0
                movsd
                .LC3(%rip),
%xmm2
                divsd
                %xmm2,
%xmm0
                addsd
                %xmm1,
%xmm0
                movsd
                %xmm0, -
56(%rbp)
                movl
                -72(%rbp),
%eax
                leal
                -1(%rax),
%edx
                movl
                -72(%rbp),
%eax
                subl
                $1, %eax
                cltq
                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
                .LC4(%rip),
%rdi
                movl
                $0, %eax
                call
                printf@PLT
                movq
                -56(%rbp),
%rax
                movq
                %rax, -
104(%rbp)
                movsd
                -104(%rbp),
%xmm0
                leaq
                .LC5(%rip),
%rdi
                movl
                leave
                .cfi_def_cfa 7,
8
                ret
                .cfi_endproc
.LFE6:
                .size
                main, .-main
                .section
                .rodata
                .align 8
.LC2:
                .long
                0
                .long
                1104006501
                .ident
                "GCC: (Ubuntu
7.3.0-16ubuntu3) 7.3.0"
                .section
                .note.GNU-
stack,"",@progbits
                $2, %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
                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)
                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
                addl
                $3, %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
                addl
                $3, %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
```

```
                $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_cf
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
```

```
                (%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
```

```
                    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
```

		%rax, %rdx movq %rdx, %rax cvtsi2sdq %rax, %xmm0 movsd .LC2(%rip), %xmm2 divsd %xmm2, %xmm0 addsd %xmm1, %xmm0 movsd %xmm0, - 72(%rbp) movl -108(%rbp), %eax leal -1(%rax), %edx movl -108(%rbp), %eax subl $1, %eax cltq 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), %xmm0 leaq .LC4(%rip), %rdi movl $1, %eax call printf@PLT movsd -72(%rbp), %xmm0 subsd -80(%rbp), %xmm0 movq -72(%rbp), %rax movapd %xmm0, %xmm1 movq %rax, - 136(%rbp) movsd -136(%rbp), %xmm0 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 .cfi_endproc .LFE6: .size main, .-main .section .rodata .align 8

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

B) CÓDIGO FIGURA 1:

CAPTURA CÓDIGO FUENTE: figura1-original.c

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

#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 desenrrollado de bucle para 4 elementos.

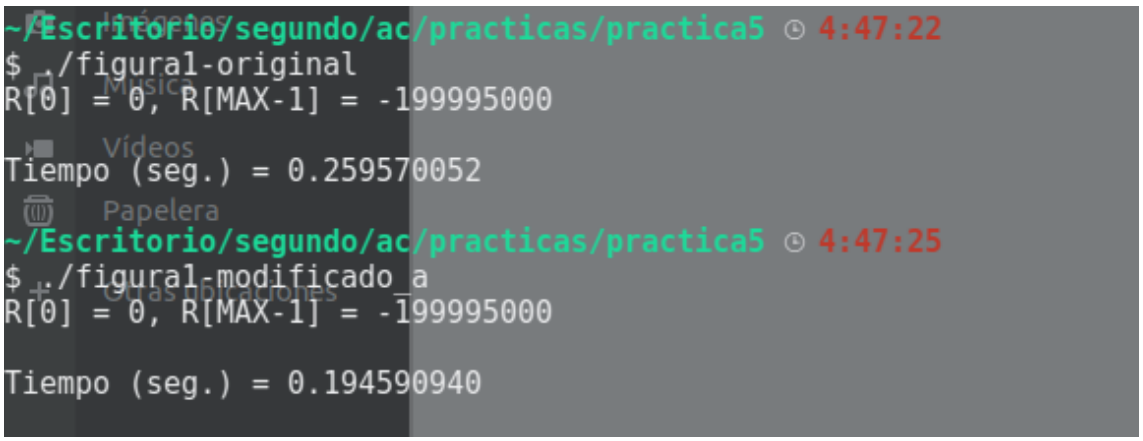
1.1. CÓDIGOS FUENTE MODIFICACIONES

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):

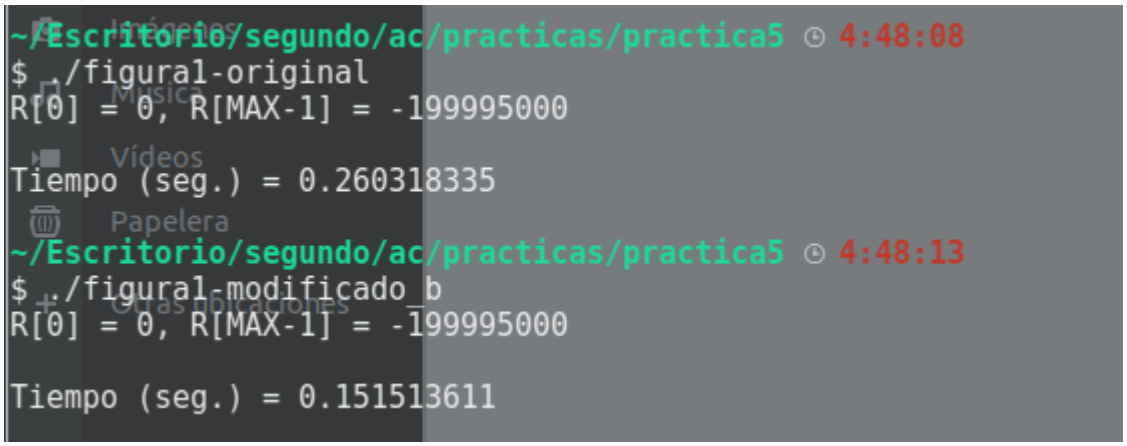


b) Captura figura1**`-modificado_b.c`**

```
#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):



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 desenrrollado 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
.file "figura1-	.file "figura1-	.file "figura1-

```
original.c"
                .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
```

```
modificado_a.c"
                .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
```

```
modificado_b.c"
                .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
```

```
                leaq    4+s(%rip), %rax
                movl    (%rdx,%rax),
%edx
                movl    %edx, %eax
                addl    %eax, %eax
                addl    %edx, %eax
                subl    -160072(%rbp),
%eax
                addl    %eax, -
160060(%rbp)
                addl    $1, -
160068(%rbp)
.L5:
                cmpl    $4999, -
160068(%rbp)
                jle     .L6
                movl    -160064(%rbp),
%eax
                cmpl    -160060(%rbp),
%eax
                jge     .L7
                movl    -160072(%rbp),
%eax
                cltq
                movl    -160064(%rbp),
%edx
                movl    %edx, -
160016(%rbp,%rax,4)
                jmp     .L8
.L7:
                movl    -160072(%rbp),
%eax
                cltq
                movl    -160060(%rbp),
%edx
                movl    %edx, -
160016(%rbp,%rax,4)
.L8:
                addl    $1, -
160072(%rbp)
.L2:
                cmpl    $40000, -
160072(%rbp)
                jle     .L9
                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   .LC0(%rip),
%xmm2
                divsd   %xmm2, %xmm0
                addsd   %xmm1, %xmm0
                movsd   %xmm0, -
160056(%rbp)
                movl    -20(%rbp), %edx
                movl    %eax, -
160060(%rbp)
                addl    $1, -
160068(%rbp)
.L3:
                cmpl    $4999, -
160068(%rbp)
                jle     .L4
                movl    -160064(%rbp),
%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   .LC0(%rip),
%xmm2
                divsd   %xmm2, %xmm0
                addsd   %xmm1, %xmm0
                movsd   %xmm0, -
160056(%rbp)
                movl    -20(%rbp), %edx
                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    %eax, -
160060(%rbp)
                movl    -160068(%rbp),
%eax
                addl    $1, %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
                addl    $1, %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    %eax, -
160060(%rbp)
                movl    -160068(%rbp),
%eax
                addl    $2, %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
                addl    $2, %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    %eax, -
160060(%rbp)
                movl    -160068(%rbp),
%eax
                addl    $3, %eax
                cltq
                leaq    0(,%rax,8),
%rdx
```

```
                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
                .L11
                call
__stack_chk_fail@PLT
.L11:
                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) 7.3.0"
                .section
                .note.GNU-
stack,"",@progbits
```

```
                $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) 7.3.0"
                .section
                .note.GNU-
stack,"",@progbits
```

```
                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
                addl
                $3, %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
                %eax, -
160060(%rbp)
                addl
                $4, -
160068(%rbp)
.L3:
                cmpl
                $4999, -
160068(%rbp)
                jle
                .L4
                movl
                -160064(%rbp),
%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
```

		.LC0(%rip), %xmm2
		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) 7.3.0"
		.section .note.GNU-stack,"",@progbits

2. El benchmark Linpack ha sido uno de los programas más ampliamente utilizados para evaluar las prestaciones de los computadores. De hecho, se utiliza como base en la lista de los 500 computadores más rápidos del mundo (el Top500 Report). El núcleo de este programa es una rutina denominada DAXPY (***D****ouble precision- real* ***A****lpha* ***X*** ***P****lus* ***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 acarrean. Incorpore los códigos al cuaderno de prácticas y destaque las diferencias entre ellos.

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

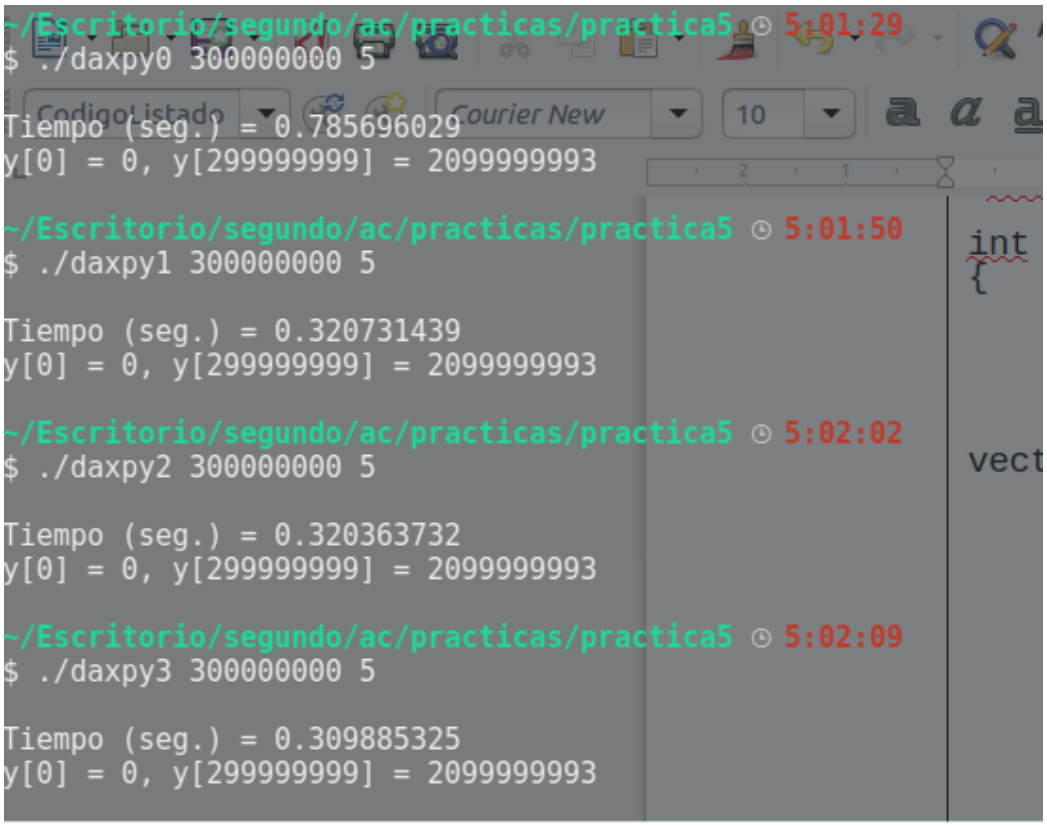
valor de la velocidad pico (Rpico) del procesador (consulte en [4] el número de ciclos por instrucción punto flotante para la familia y modelo de procesador que está utilizando) y compárela con el valor obtenido para Rmax. -Consulte la Lección 3 del Tema 1.

CAPTURA CÓDIGO FUENTE: daxpy.c

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

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 [tamanio
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.	0.785696029	0.320731439	0.320363732	0.309885325

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