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

Doble Grado Ing. Inform. y Mat.

# Arquitectura de Computadores (AC)

# Cuaderno de prácticas. Bloque Práctico 4. Optimización de código

Estudiante (nombre y apellidos): 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) <u>No use máquinas virtuales.</u> 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 1scpu en la máquina en la que ha tomado las medidas

- 1. Para el núcleo que se muestra en el Figura 1, y para un programa que implemente la multiplicación de matrices (use variables globales):
  - 1.1 Modifique el código C para reducir el tiempo de ejecución del mismo. Justifique los tiempos obtenidos (use −O2) a partir de la modificación realizada. Incorpore los códigos modificados en el cuaderno.

- 1.2 Genere los códigos en ensamblador con -O2 para el original y dos códigos modificados obtenidos en el punto anterior (incluido el que supone menor tiempo de ejecución) e incorpórelos al cuaderno de prácticas. Destaque las diferencias entre ellos en el código ensamblador.
- 1.3 (Ejercicio EXTRA) Intente mejorar los resultados obtenidos transformando el código ensamblador del programa para el que se han conseguido las mejores prestaciones de tiempo

**Figura 1**. Código C++ que suma dos vectores

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

main()
{
    for (ii=0; ii<40000;ii++) {
        X1=0; X2=0;
        for(i=0; i<5000;i++) X1+=2*s[i].a+ii;
        for(i=0; i<5000;i++) X2+=3*s[i].b-ii;
        if (X1<X2) R[ii]=X1 else R[ii]=X2;
    }
}</pre>
```

# A) MULTIPLICACIÓN DE MATRICES:

CAPTURA CÓDIGO FUENTE: pmm-secuencial.c

```
#include <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
         (argc < 2) {
  fprintf(stderr, "Falta el tamanio de la matriz\n");</pre>
          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:
                 1 2 3
2 3 4
            b=
                 3
                    4
                 2
                    3
            c=
            a= 20 26 32
29 38 47
       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 (j=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
       //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 <stdib.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
(argc < 2) {
  fprintf(stderr, "Falta el tamanio de la matriz\n");
  exit(-1);</pre>
       int N = atoi(argv[1]);
       if(N>MAX) N = MAX;
       int i, j,k;
       struct timespec cgt1,cgt2, cgt3; double ncgt,ncgt2;
       //Inicializacion
       //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 columas
for(ina);
        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;
}
```

# 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");</pre>
             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);
```

```
~/Escritorio/segundo/ac/practicas/practica5 ⊙ 4:23:19
$ gcc -02 pmm-modificado_2.c -o pmm-modificado_2
Papelera
~/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
Relacion
Tiempo (seg.) solo multiplicacion = 0.000000963
Tiempo (seg.) con trasposicion de matriz = 0.000000579 segundos mas
```

1.1. TIEMPOS: para matrices de tamaño 1500 x 1500

Modificación	-O0	-01	-O2
Sin modificar	21.33872346	4.992951580	4.863861247
Modificación	13.843146801	2.896383608	3.163519620
a)			
Modificación	13.184861515	2.188715671	2.221235431
b)			

#### 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-secuenci	ial.s	pmm-secuenc: modificado_l	ial- b.s	pmm-secuenci modificado_d	ial- c.s
secuencial.c"	.file "pmm- .text .comm	modificado.c"	.file "pmm- .text .comm a,36000000,32 .comm	modificado_2.c"	.file "pmm- .text .comm a,36000000,32 .comm
a,36000000,32	.comm		b,36000000,32 .comm c,360000000,32 .section .rodata		b,36000000,32 .comm c,360000000,32 .section .rodata
b,36000000,32	.comm	.LC0:	.string "%d "	.LC0:	.string "%d "
c,36000000,32	.section .rodata		.text .globl imprimirMatriz .type		.text .globl imprimirMatriz .typę
.LC0:	.string "%d "	@function imprimirMatriz: .LFB5:	<pre>imprimirMatriz, .cfi startproc</pre>	@function imprimirMatriz: .LFB5:	<pre>imprimirMatriz, .cfi startproc</pre>
	.text .globl		pushq %rbp .cfi_def_cfa_of		pushq %rbp .cfi_def_cfa_of
imprimirMatri	.z .type	fset 16 -16	.cfi_offset 6,	fset 16 -16	.cfi_offset 6,
imprimirMatri imprimirMatri .LFB5:		gister 6	movq %rsp, %rbp .cfi_def_cfa_re	gister 6	movq %rsp, %rbp .cfi_def_cfa_re
roc	.cfi_startp pushq %rbp .cfi_def_cf		subq \$32, %rsp movq %rdi, -24(%rbp) movl %esi, -28(%rbp) movl \$0, -8(%rbp)		subq \$32, %rsp movq %rdi, -24(%rbp) movl %esi, -28(%rbp) movl \$0, -8(%rbp)
6, -16	.cfi_offset movq %rsp, %rbp .cfi def cf	.L5:	jmp .L2 movl \$0, -4(%rbp) jmp .L3	.L5:	jmp .L2 mov1 \$0, -4(%rbp) jmp .L3
a_register 6	subq \$32, %rsp movq %rdi, -	.L4:	movl -8(%rbp), %eax cltq imulq \$12000, %rax,	.L4:	movl -8(%rbp), %eax cltq imulq \$12000, %rax,
24(%rbp)	movl %esi, -	%rdx	movq -24(%rbp), %rax addq	%rdx	movq -24(%rbp), %rax addq

7

28(%rhn)			%rax, %rdx		%rax, %rdx
28(%rbp)	movl \$0, -		movl -4(%rbp), %eax cltq		movl -4(%rbp), %eax cltq
8(%rbp)	jmp	%eax	movl (%rdx,%rax,4),	%eax	movl (%rdx,%rax,4),
.L5:	.L2	/Meax	movl %eax, %esi	/ACAX	movl %eax, %esi
	movl \$0, -	  %rdi	leaq .LC0(%rip),	%rdi	leaq .LC0(%rip),
4(%rbp)	jmp .L3	701 41	movl \$0, %eax call	701 41	movl \$0, %eax call
.L4:	_		printf@PLT addl		printf@PLT addl
0/00/	mov⊥ -8(%rbp),	.L3:	\$1, -4(%rbp) movl	.L3:	\$1, -4(%rbp) movl
%eax	cltq		-4(%rbp), %eax cmpl		-4(%rbp), %eax cmpl
%rax, %rdx	1mulq \$12000,		-28(%rbp), %eax jl .L4		-28(%rbp), %eax jl .L4
701 dx, 701 dx	movq -24(%rbp),		movl \$10, %edi call		movl \$10, %edi call
%rax	addq		putchar@PLT addl \$1, -8(%rbp)		putchar@PLT addl \$1, -8(%rbp)
	%rax, %rdx movl	.L2:	movl	.L2:	movl
%eax	-4(%rbp),		-8(%rbp), %eax cmpl -28(%rbp), %eax		-8(%rbp), %eax cmpl -28(%rbp), %eax
	cltq movl		jl .L5 movl		jl .L5 movl
%rax,4), %eax			\$10, %edi call putchar@PLT		\$10, %edi call
	movl %eax, %esi		nop leave		putchar@PLT nop leave
%rdi	leaq .LCO(%rip),	8	.cfi_def_cfa 7, ret	8	.cfi_def_cfa 7, ret
701 01	movl \$0, %eax	.LFE5:	.cfi_endproc .size	.LFE5:	.cfi_endproc .size
	call printf@PLT	imprimirMatriz	imprimirMatriz,	imprimirMatriz	imprimirMatriz,
	addl \$1, -		.section .rodata .align 8		.section .rodata .align 8
4(%rbp)  .L3:		.LC1:	.string "Falta el	.LC1:	.string "Falta el
	movl -4(%rbp),	tamanio de la mat .LC3:	.1 12 \11	tamanio de la mai .LC3:	L1 17 / II
%eax	cmpl	nA[N-1][N-1]=%d\r	"\nA[0][0]=%d\ ""	nA[N-1][N-1]=%d\i	"\nA[0][0]=%d\ n"
%eax	-28(%rbp), jl	.LC4:	string	.LC4:	.align 8 .string _"\nTiempo
	.L4 movl	(seg.) solo multi %11.9f\n"	"\nTiempo iplicacion =	(seg.) solo mult: %11.9f\n"	"\nTiempo iplicacion =
	\$10, %edi call	.LC5:	.align 8 .string	.LC5:	.align 8 .string
	putchar@PLT addl	(seg.) con traspo	"\nTiempo	(seg.) con traspo	"\nTiempo
8(%rbp)	\$1, -	matriz = %11.9f\t mas\n"	.text	matriz = %11.9f\ mas\n"	.text_
.L2:	movl		.globl main .type		.globl main .type
%eax	-8(%rbp),	main: .LFB6:	main, @function	main: .LFB6:	main, @function
%pay	cmpl -28(%rbp),		.cfi_startproc pushq		.cfi_startproc pushq
%eax	jl .L5	fset 16	%rbp' .cfi_def_cfa_of	fset 16	%rbp .cfi_def_cfa_of
	movl \$10, %edi	-16	.cfi_offset 6,	-16	.cfi_offset 6,
	call putchar@PLT	gistor 6	%rsp, %rbp .cfi_def_cfa_re	gistor 6	%rsp, %rbp .cfi_def_cfa_re
	nop leave	gister 6	subq \$144, %rsp	gister 6	subq \$144, %rsp
a 7, 8	.cfi_def_cf	116(%rbp)	movl %edi, -	116(%rbp)	movl %edi, -
	ret .cfi_endpro	128(%rbp)	movq %rsi, -	128(%rbp)	movq %rsi, -
c .LFE5:	0.170		movq %fs:40, %rax	(,w, bp)	movq %fs:40, %rax
imprimirMotri	.S1Ze		movq %rax, -8(%rbp) xorl		movq %rax, -8(%rbp) xorl
imprimirMatri imprimirMatri			%eax, %eax cmpl \$1, -116(%rbp)		%eax, %eax cmpl \$1, -116(%rbp)
	.rodata .align 8		jg .L7 movq		jg .L7 movq
<u>R</u>	- wgii 0		· · · · · · · · · · · · · · · · · · ·	itoetura v Toenologís	<u> </u>

		1		T	
.LC1:	etrina	%rax	stderr(%rip),	%rax	stderr(%rip),
	.string "Falta el	1.00	movq %ray %rcy		movq %rax, %rcx
tamanio de la	a matriz\n"		%rax, %rcx mov1		mov1
.LC2:			\$30, %edx mov1		\$30, %edx mov1
	.string "c[0][0] -		\$1, %esi		\$1, %esi
%d, c[N][N] =	"c[0][0] = = %d\n"		leaq .LC1(%rip),		leáq .LC1(%rip),
LC4:		%rdi	call	%rdi	call
	.string "\nA[0][0]=		fwrite@PLT		fwrite@PLT
%d\nA[N-1][N-	.11=%d\n"		movl \$-1, %edi		movl \$-1, %edi
.LC5:			call exit@PLT		call exit@PLT
	.string "\nTiempo	.L7:	movq	.L7:	movq
(seg.) = %11.	.9f\n"		-128(%rbp),		-128(%rbp),
(339.)	.text	%rax	adda	%rax	adda
	.globl		addq \$8, %rax movq		addq \$8, %rax movq
	main .type		(%rax), %rax		(%rax), %rax
	main,		movq %rax, %rdi		movq %rax, %rdi
@function			call atoi@PLT		call atoi@PLT
main:  .LFB6:			movl		mov1
121 001	.cfi_startp	100(%rbp)	%eax, -	108(%rbp)	%eax, -
roc			cmpl \$3000, -		cmpl \$3000, -
	pushq %rbp	100(%rbp)	,	108(%rbp)	
	.cfi_def_cf		jle .L8		jle .L8
a_offset 16	ofi offert		movl \$3000, -		movl \$3000, -
6, -16	.cfi_offset	100(%rbp) .L8:	•	108(%rbp) .L8:	,
, 10	movq	. 201	movl		movl
	%rsp, %rbp_		\$0, -96(%rbp) jmp .L9		\$0, -104(%rbp) jmp .L9
a_register 6	.cfi_def_cf	.L12:	. L9	.L12:	. L9
· · g · · · ·	subg		movl \$0, -92(%rbp)		movl \$0, -100(%rbp)
	\$112, %rsp movl		jmp .L10		jmp .L10
	%edi, -	.L11:	_	.L11:	_
84(%rbp)	,		movl -92(%rbp), %eax		movl -100(%rbp),
	movq %rsi, -		cltq mov1	%eax	cltq
96(%rbp)	70. 027		-96(%rbp), %edx		movľ -104(%rbp),
	movq %fs:40,		movslq %edx, %rdx	%edx	
%rax	7013.40,		imulq \$3000, %rdx,		movslq %edx, %rdx
	movq	%rdx	addq		imulq \$3000, %rdx,
8(%rbp)	%rax, -		%rdx, %rax leag	%rdx	addo
(701.56)	xorl	%rdx	leaq 0(,%rax,4),		addq %rdx, %rax leaq
	%eax, %eax	701 GX	leaq a(%ṛip), %rax	0/mdv	0(,%rax,4),
	cmp⊥ \$1, -		mov I	%rdx	leaq
84(%rbp)	•		\$0, (%rdx,%rax) mov1		a(%rip), %rax movl
	jg .L7		-96(%rbp), %edx		mov1 \$0, (%rdx,%rax) mov1
	movq		-92(%rbp), %eax addl	%edx	-104(%rbp),
stderr(%rip),			%edx, %eax	/ucu^	mov1
Scucii (MITH),	mova		leal 1(%rax), %ecx	%eax	-100(%rbp),
	%rax, %rcx		mòvl -92(%rbp), %eax		addl %edx, %eax
	movl \$30, %edx		cltà movl		leal' 1(%rax), %ecx
	mo∨⊥		-96(%rbp), %edx		mov1 -100(%rbp),
	\$1, %esi		movslq %edx, %rdx imulq	%eax	` ' ' ' ' '
	leaq .LC1(%rip),		1mu1q \$3000, %rdx,		cltq movl
%rdi		%rdx		%edx	-104(%rbp),
	call fwrite@PLT		addq %rdx, %rax leaq		movslq %edx, %rdx
	movl	%rdx	0(,%rax,4),		imulq \$3000, %rdx,
	\$-1, %edi call	/01 U.A	leaq b(%rip), %rax	%rdx	, , , , , , , , , , , , , , , , , , ,
	exit@PLT		IIIOAT		addq %rdx, %rax
.L7:	-	%rax)	%ecx, (%rdx,		leaq 0(,%rax,4),
	movq -96(%rbp),	·	movl -96(%rbp), %eax	%rdx	leaq
%rax			leal 2(%rax), %edx		b(%rip), %rax movl
	addq \$8, %rax		mòvl '	0()	%ecx, (%rdx,
	mova		-92(%rbp), %eax leal	%rax)	movl
0/1501/	(%rax),	%ecx	(%rdx,%rax),	%eax	-104(%rbp),
%rax	movq		movl -92(%rbp), %eax		leal 2(%rax), %edx
	%rax, %rdi		cltq movl		movl -100(%rbp),
	call atoi@PLT		-96(%rbp), %edx	%eax	
	a cot@FLI		movslq		leal

	movl %eax, -		%edx, %rdx imulq \$3000, %rdx,	%ecx	(%rdx,%rax), movl
72(%rbp)	cmp1	%rdx	addq %rdx, %rax	%eax	-100(%rbp),
72(%rbp)	\$3000, -		%rdx, %rax leaq 0(,%rax,4),		cltq movl -104(%rbp),
	jle .L8	%rdx	leaq c(%rip), %rax	%edx	movslq %edx, %rdx
	movl \$3000, -		movl %ecx, (%rdx,		%edx, %rdx imulq \$3000, %rdx,
72(%rbp) .L8:	,	%rax)	addl \$1, -92(%rbp)	%rdx	addq %rdx, %rax
	movl \$0, -	.L10:	\$1, -92(%rbp) movl		%rdx, %rax leaq 0(,%rax,4),
68(%rbp)			-92(%rbp), %eax	%rdx	leaq c(%rip), %rax
.L12:	jmp .L9	%eax	-100(%rbp), jl		movl %ecx, (%rdx,
	movl \$0, -		.L11 addl	%rax)	addl \$1, -100(%rbp)
64(%rbp)	imp	.L9:	\$1, -96(%rbp) movl	.L10:	mov1
.L11:	. L'10		-96(%rbp), %eax cmpl	%eax	-100(%rbp),
	movl -64(%rbp),	%eax	-100(%rbp), jl	%eax	cmpl -108(%rbp),
%eax	cltg		.L12 leaq -32(%rbp), %rax		jl .L11 addl
	movl -68(%rbp),		movq %rax, %rsi	.L9:	\$1, -104(%rbp)
%edx	movslq		movl \$0, %edi call	%eax	movl -104(%rbp),
	%edx, %rdx imulq	clock_gettime@PL	г _		cmpl -108(%rbp),
%rdx, %rdx	\$3000,		movl \$0, -96(%rbp) jmp	%eax	jl .L12
701 dx, 701 dx	addq %rdx, %rax	.L16:	.L13		leaq -32(%rbp), %rax
	leaq 0(,%rax,4),		movl \$0, -92(%rbp) jmp		%rax, %rsi movl
%rdx	leaq .	.L15:	.L14		\$0, %edi call
%rax	a(%rip),		movl -92(%rbp), %eax cltq	clock_gettime@PL1	mov1
701 47	movl \$0, (%rdx,		movi -96(%rbp), %edx movslq %edx, %rdx		\$0, -104(%rbp) jmp .L13
%rax)	movl		imuiq	.L16:	movl
%edx	-68(%rbp),	%rdx	\$3000, %rdx, addq		\$0, -100(%rbp) jmp .L14
70CUX	movl -64(%rbp),		%rdx, %rax leag	.L15:	movl
%eax	addl	%rdx	0(,%rax,4), leaq	%eax	-100(%rbp), cltq
	%edx, %eax leal		c(%rip), %rax movl	or at	movi -104(%rbp),
%ecx	1(%rax),	%eax	(%rdx,%rax), movl	%edx	movslq %edx. %rdx
70ECX	movl -64(%rbp),		%eax, -84(%rbp) movl	0/m day	movslq %edx, %rdx imulq \$3000, %rdx,
%eax			-96(%rbp), %eax cltq movl	%rdx	addq %rdx, %rax
	cltq movl -68(%rbp),		-92(%rbp), %edx movslq %edx, %rdx	%rdx	leaq 0(,%rax,4),
%edx	movela		imulq \$3000, %rdx,	/// u/	leaq c(%rip), %rax
	movslq %edx, %rdx imula	%rdx	addq %rdx, %rax	%eax	mòvl (%rdx,%rax),
%rdx, %rdx	imulq \$3000,	ovd.	leaq 0(,%rax,4),	, and an	movl %eax, -84(%rbp)
% ux, % ux	addq %rdx, %rax	%rdx	leaq c(%rip), %rax	%eax	movl -104(%rbp),
	leaq 0(,%rax,4),	0/- 4	movl (%rdx,%rax),	, sean	cltq movl
%rdx		%edx	movl -92(%rbp), %eax	%edx	-100(%rbp), movslq
%ray	leaq b(%rip),		cltq mov1		movslq %edx, %rdx imulq \$3000 %rdy
%rax	movl		-96(%rbp), %ecx movslq %ecx, %rcx	%rdx	\$3000, %rdx, addq
(%rdx,%rax)	%ecx,	%rcx	imulq \$3000, %rcx,		addq %rdx, %rax leaq 0(,%rax,4),
%02Y	movl -68(%rbp),	701 CA	addq %rcx, %rax	%rdx	leaq c(%rip), %rax
%eax	leal	%rcx	leaq′ 0(,%rax,4),		c(%rip), %rax movl (%rdx,%rax),
%edx	2(%rax),	,	leaq c(%rip), %rax movl	%edx	movl
10	movl			itoetura y Toenología	-100(%rbp),

	2.1/2: 1.3	T	A. 1. (**)	La.	
%eax	-64(%rbp),	%rax)	%edx, (%rcx,	%eax	cltg
/ocax	leal		movl -96(%rbp), %eax		movľ -104(%rbp),
%rax), %ecx	(%rdx,		cltà ''' movl	%ecx	movslq %ecx, %rcx
// ax ), //ccx	movl		-92(%rbp), %edx		%ecx, %rcx imulq
0/004	-64(%rbp),		movšlq %edx, %rdx	%rov	\$3000, %rcx,
%eax	cltq		imulq \$3000, %rdx,	%rcx	addq
	movľ	%rdx	addq		%rcx, %rax leaq
%edx	-68(%rbp),		%rdx, %rax leaq	%rcx	0(,%rax,4),
700 471	movslq	%rcx	leaq' 0(,%rax,4),		leaq c(%rip), %rax
	%edx, %rdx imulq		leaq c(%rip), %rax		mòvl // %edx, (%rcx,
	\$3000,		mov1 -84(%rbp), %edx	%rax)	movl
%rdx, %rdx	addq		mov1 %edx, (%rcx,	%eax	-104(%rbp),
	%rdx, %rax	%rax)	, , ,	/0Cax	cltq
	leaq 0(,%rax,4),	144.	addl \$1, -92(%rbp)	0/	movl -100(%rbp),
%rdx	0(, %1 ax, 4),	.L14:	movl	%edx	movslq %edx, %rdx
	leaq		-92(%rbp), %eax cmpl		
%rax	c(%rip),	%eax	-100(%rbp),	%rdx	\$3000, %rdx,
	movl		jl .L15		addq %rdx, %rax
(%rdx,%rax)	%ecx,		addl \$1, -96(%rbp)		leaq 0(,%rax,4),
\	addl	.L13:	_	%rcx	
64(%rbp)	\$1, -		movl -96(%rbp), %eax		leaq c(%rip), %rax
.L10:	_	04.5	cmpl -100(%rbp),		-84(%rbp), %edx
	movl -64(%rbp),	%eax	jl		movl %edx, (%rcx,
%eax	- O+(/01 DP),		.L16	%rax)	addl
	cmpl -72(%rbp),		leaq -64(%rbp), %rax movq	.L14:	\$1, -100(%rbp)
%eax			%rax, %rsi movl		movl -100(%rbp),
	jl		\$0, %edi call	%eax	cmpl
	.L11 addl	clock_gettime@PL		%eax	-108(%rbp),
60 ( % rbp )	\$1, -	CIOCK_gettime@FL	mov1	//deax	jl .L15
68(%rbp)  .L9:			\$0, -96(%rbp) jmp		addl
	movl	.L22:	.L17	.L13:	\$1, -104(%rbp)
%eax	-68(%rbp),		movl \$0, -92(%rbp)		movl -104(%rbp),
	cmpl		jmp .L18	%eax	cmpl
%eax	-72(%rbp),	.L21:	movl	%eax	-108(%rbp),
70000	j1		\$0, -88(%rbp) jmp		jl .L16
	.L12 movl	.L20:	.̃L19		leaq -64(%rbp), %rax
04 -	-72(%rbp),		movl -92(%rbp), %eax		movq %rax, %rsi
%eax	leal		cltq		movT
041	-1(%rax),		mov1 -96(%rbp), %edx		\$0, %edi call
%edx	movl		movslq %edx, %rdx	clock_gettime@PL <sup>-</sup>	
	-72(%rbp),	0/11-11-1	imulq \$3000, %rdx,	0/	movl -108(%rbp),
%eax	subl	%rdx	addq	%eax	leal
	\$1, %eax		%rdx, %rax leaq		3(%rax), %edx testl
	clta	%rdx	0(,%rax,4),		%eax, %eax cmovs
	movslq %edx, %rdx		leaq a(%rip), %rax		%edx, %eax
	imulq \$3000,		movl (%rdx,%rax),		\$2, %eax mov1
%rdx, %rdx		%edx	mov1		%eax, -88(%rbp)
	addq %rdx, %rax		-88(%rbp), %eax		\$0, -104(%rbp) jmp
	leaq		movl -96(%rbp), %ecx	124.	. L17
%rdx	leaq 0(,%rax,4),		movslq %ecx, %rcx	.L24:	movl
701 U.A	leaq		imuiq		\$0, -100(%rbp) jmp
%rav	c(%rip),	%rcx	\$3000, %rcx,	.L23:	. L18
%rax	movl		addq %rcx, %rax		movl \$0, -96(%rbp) movl
%ray) %adv	(%rdx,		leaq 0(,%rax,4),		\$0, -92(%rbp)
%rax), %edx	movl	%rcx			jmp .L19
%02Y	c(%rip),		leaq b(%rip), %rax movl	.L20:	movl
%eax	movl	%ecx	(%rcx,%rax),	%eax	-100(%rbp),
	%eax, %esi		movl -88(%rbp), %eax		cltq movl
	leaq .LC2(%rip),		cltg	%pdv	-104(%rbp),
%rdi	- ·(·-· <b>-</b> P//		mov1 -92(%rbp), %esi	%edx	movslq
				uitectura v Tecnologí	

	movl \$0,_%eax		movslq %esi, %rsi		%edx, %rdx imulq
	call printf@PLT	%rsi	imulq \$3000, %rsi,	%rdx	\$3000, %rdx,
	leaq -48(%rbp),		addq %rsi, %rax		addq %rdx, %rax leag
%rax		  %rsi	leaq 0(,%rax,4),	%rdx	0(,%rax,4),
	movq %rax, %rsi	701 31	leaq c(%rip), %rax		leaq a(%rip), %rax movl
	movl \$0, %edi call		mòvl (%rsi,%rax),	%edx	(%rdx,%rax),
		%eax	imull		movl -96(%rbp), %eax cltg
clock_gettime	e@PLT movl		%ecx, %eax leal (%rdx,%rax),		movl -104(%rbp),
68(%rbp)	\$0, -	%ecx	movl	%ecx	movslq %ecx, %rcx
00(/// 00)	jmp .L13		-92(%rbp), %eax cltq movl		imulq \$3000, %rcx,
.L18:			-96(%rbp), %edx movslq %edx, %rdx	%rcx	addg
0.4 (0) (1)	movl \$0, -		%edx, %rdx imulq \$3000, %rdx,		%rcx, %rax leaq 0(,%rax,4),
64(%rbp)	jmp	%rdx	addq	%rcx	leaq
.L17:	.L14		%rdx, %rax leag		b(%rip), %rax movl
	movl \$0, -	%rdx	0(,%rax,4), leaq	%ecx	(%rcx,%rax), movl
60(%rbp)			a(%rip), %rax movl		-96(%rbp), %eax cltg
.L16:	jmp .L15	%rax)	%ecx, (%rdx, addl	%esi	movl -100(%rbp),
	movl	.L19:	\$1, -88(%rbp)	/vesi	movslq %esi, %rsi
%eax	-64(%rbp),		movl -88(%rbp), %eax		imulq \$3000, %rsi,
	cltq movl	%eax	cmpl -100(%rbp),	%rsi	addq %rsi, %rax
%edx	-68(%rbp),	/acux	jl .L20		leaq 0(,%rax,4),
	movslq %edx, %rdx	140.	addl \$1, -92(%rbp)	%rsi	leaq c(%rip), %rax
	imulq \$3000,	.L18:	movl -92(%rbp), %eax		movl (%rsi,%rax),
%rdx, %rdx	addq	0/	cmpl -100(%rbp),	%eax	imull
	%rdx, %rax leaq	%eax	jl .L2 <u>1</u>		%ecx, %eax leal (%rdx,%rax),
%rdx	0(,%rax,4),		addl \$1, -96(%rbp)	%ecx	movl
761 U.X	leaq	.L17:	movl -96(%rbp), %eax	%eax	-100(%rbp), cltg
%rax	a(%rˈip),		cmpl -100(%rbp),		movi -104(%rbp),
	movl (%rdx,	%eax	jl .L22	%edx	movslq %edx, %rdx
%rax), %edx	movl		leaq -48(%rbp), %rax		imulq \$3000, %rdx,
%eax	-60(%rbp),		%rax, %rsi	%rdx	addq %rdx, %rax
	cltq movl		movl \$0, %edi call		leaq 0(,%rax,4),
%ecx	-68(%rbp),	clock_gettime@PL	Г	%rdx	leaq a(%rip), %rax
	movslq %ecx, %rcx		movq -48(%rbp), %rdx movq		movl %ecx, (%rdx,
	imulq \$3000,		-64(%rbp), %rax subq	%rax)	movl
%rcx, %rcx	addq		%rax, %rdx movq %rdx,_%rax	%eax	-100(%rbp), cltg
	%rcx, %rax leaq		cvtsi2sdq %rax, %xmm1	0/adv	movl -104(%rbp),
%rcv	0(,%rax,4),		movq -40(%rbp), %rdx movq	%edx	movslq %edx, %rdx
%rcx	leaq		-56(%rbp), %rax subg	O(m do.	imulq \$3000, %rdx,
%rax	b(%rip),		%rax, %rdx movq %rdx. %rax	%rdx	addq %rdx, %rax
0(10.014)	movl (%rcx,		%rdx, %rax cvtsi2sdq %rax, %xmm0 movsd	O(m do.	leaq 0(,%rax,4),
%rax), %ecx	movl	%xmm2	movsd .LC2(%rip),	%rdx	leaq a(%rip), %rax
%eax	-64(%rbp),		divsd %xmm2, %xmm0	a	movl (%rdx,%rax),
	cltq movl		addsd′ %xmm1, %xmm0 movsd	%edx	movl -96(%rbp), %eax
%esi	-60(%rbp),	80(%rbp)	%×mm0, -		addi \$1, %eax
	movslq %esi, %rsi		movq -48(%rbp), %rdx		cltq movl
	imulq		movq -32(%rbp), %rax	%ecx	-104(%rbp),
17				itactura y Tacnalagía	

%rsi, %rsi	\$3000,		subq %rax, %rdx		movslq %ecx, %rcx
701 31, 701 31	addq		movq %rdx, %rax		imulq \$3000, %rcx,
	%rsi, %rax		cvtsi2sdq %rax, %xmm1	%rcx	addq
	leaq 0(,%rax,4),		movq -40(%rbp), %rdx		%rcx, %rax
%rsi	-		movq		leaq′ 0(,%rax,4),
	leaq		-24(%rbp), %rax subq	%rcx	leaq
%rax	c(%rip),		%rax %rdx		b(%rip), %rax movl
1.0.	movl.		%rdx, %rax	0/	(%rcx,%rax),
%rax), %eax	(%rsi,		movq %rdx, %rax cvtsi2sdq %rax, %xmm0 movsd	%ecx	movl
701 ax / / 70cax	imull		movsd .LC2(%rip),		-96(%rbp), %eax addl
	%ecx, %eax	%xmm2	divsd		\$1, %eax cltg
	leal (%rdx,		%xmm2, %xmm0 addsd		movi -100(%rbp),
%rax), %ecx	_		%xmm1, %xmm0	%esi	
	movl -64(%rbp),		movsd %xmm0, -		movslq %esi, %rsi
%eax		72(%rbp)	movl		imulq \$3000, %rsi,
	cltq	%eax	-100(%rbp),	%rsi	addq
	movl -68(%rbp),		leal -1(%rax), %edx		%rsi, %rax leaq
%edx			movl '	0/moi	0(,%rax,4),
	movslq %edx, %rdx	%eax	-100(%rbp),	%rsi	leaq c(%rip), %rax
	ımu⊥q		subl \$1, %eax cltg		c(%rip), %rax movl
0/rdy 0/rdy	\$3000,		cltq	  %eax	(%rsi,%rax),
%rdx, %rdx	addq		movslq %edx, %rdx imulq	/ ACCUA	imull
	%rdx, %rax	ny sal	\$3000, %rdx,		%ecx, %eax leal
	leaq´ 0(,%rax,4),	%rdx	addq	%ecx	(%rdx,%rax),
%rdx	θ(, %i αλ, 4),		%rdx, %rax leag		movl -100(%rbp),
	leaq	%rdx	leaq′ 0(,%rax,4),	%eax	cltq
%rax	a(%rˈip),	701 U.X	leaq		movl -104(%rbp),
70. 67.	movl		a(%rip), %rax movl	%edx	
(%rdx,%rax)	%ecx,	%edx	(%rdx,%rax),		movslq %edx, %rdx
(701 UX, 701 UX)	addl		movl a(%ṛip), %eax		imulq \$3000, %rdx,
60(%/shn)	\$1, -		movl %eax, %esi	%rdx	addq
60(%rbp) .L15:			leaq .LC3(%rip),		%rdx, %rax
	movl	%rdi	_	ov - d	leaq′ 0(,%rax,4),
%eax	-60(%rbp),		movl \$0, %eax call	%rdx	leaq
/ocax	cmpl		call printf@PLT		a(%rip), %rax movl
9/004	-72(%rbp),		movq -80(%rbp), %rax	%rax)	%ecx, (%rdx,
%eax	jl		movq %rax, -	, and	movl -100(%rbp),
	.L16	136(%rbp)		%eax	_
	addl \$1, -		movsd -136(%rbp),		cltq movl
64(%rbp) .L14:	Ψ=/	%×mm0	leaq	%edx	-104(%rbp),
.L14:	movl	%rdi	.LC4(%rip),		movslq %edx, %rdx
	-64(%rbp),		movl \$1, %eax call		imulq \$3000, %rdx,
%eax	_		call printf@PLT	%rdx	
	cmpl -72(%rbp),		movsd		addq %rdx, %rax
%eax		%×mm0	-72(%rbp),	Or and a	leaq 0(,%rax,4),
	jl .L17		subsd -80(%rbp),	%rdx	leag
	addl	%×mm0			a(%rˈip), %rax movl
68(%rbn)	\$1, -		movq -72(%rbp), %rax movapd	%edx	(%rdx,%rax),
68(%rbp) .L13:			%xmm0, %xmm1	, sour	movl -96(%rbp), %eax
	movl	120(0/5)	movq %rax, -		add1
%eax	-68(%rbp),	136(%rbp)	movsd		\$2, %eax cltq
	cmpl	%×mm0	-136(%rbp),		movľ -104(%rbp),
%eax	-72(%rbp),		leag .LC5(%rip),	%ecx	movsla
70001	jl .	%rdi	movil		movslq %ecx, %rcx imulq
	.L18		\$2, %eax call	%rcv	\$3000, %rcx,
	leaq -32(%rbp),		printr@PLI	%rcx	addq
%rax			movl \$0, %eax		%rcx, %rax leaq
	movq %rax, %rsi		movq -8(%rbp), %rdi	%rcx	0(,%rax,4),
	mo∨⊥		xorq %fs:40, %rdi		leaq b(%rip), %rax
	\$0, %edi call		je .L24		movl (%rcx,%rax),
			call	%ecx	_
clock_gettime		stack_chk_fail(	@PLT		movl -96(%rbp), %eax
	movq	.L24:	Dant- A	 uitectura v Tecnologí	addI

%rdx	-32(%rbp), movq	8	leave .cfi_def_cfa 7,		\$2, %eax cltq movl
%rax	-48(%rbp), subq	.LFE6:	ret .cfi_endproc .size	%esi	-100(%rbp), movslq %esi, %rsi
	%rax, %rdx movq		main,main .section .rodata .align 8	%rsi	imulq \$3000, %rsi, addq
	%rdx, %rax cvtsi2sdq %rax, %xmm1	.LC2:	.long	0/	%rsi, %rax leaq 0(,%rax,4),
%rdx	movq -24(%rbp),	7.3.0-16ubuntu3)	.long 1104006501 .ident "GCC: (Ubuntu	%rsi	leaq c(%rip), %rax movl
%rax	movq -40(%rbp),	7.3.0-16ubuntu3) stack,"",@progbi	.section .note.GNU-	%eax	(%rsi,%rax), imull %ecx, %eax
	subq %rax, %rdx movq	, , , , ,		%ecx	leal' (%rdx,%rax), movl
	%rdx, %rax cvtsi2sdq %rax, %xmm0			%eax	-100(%rbp), cltg
%xmm2	movsd .LC3(%rip),			%edx	movl -104(%rbp), movslq
%×mm0	divsd %xmm2,			%rdx	%edx, %rdx imulq \$3000, %rdx,
	addsd %xmm1,			701 UX	addq %rdx, %rax leaq 0(,%rax,4),
%xmm0	movsd %xmm0, -			%rdx	leaq a(%rip), %rax
56(%rbp)	movl -72(%rbp),			%rax)	movl %ecx, (%rdx, movl
%eax	leal -1(%rax),			%eax	-100(%rbp), cltq mov1
%edx	movl -72(%rbp),			%edx	-104(%rbp), movslq %edx, %rdx
%eax	subl \$1, %eax			%rdx	\$3000, %rdx,
	cltq movslq %edx, %rdx imula				addq %rdx, %rax leaq 0(,%rax,4),
%rdx, %rdx	imulq \$3000,			%rdx	leaq a(%rip), %rax movl
,	addq %rdx, %rax leag			%edx	(%rdx,%rax),
%rdx	0(,%rax,4), leaq				-96(%rbp), %eax addl \$3, %eax cltg
%rax	a(%rip), movl			%ecx	movl -104(%rbp), movslq
%rax), %edx	(%rdx, movl			%rcx	movslq %ecx, %rcx imulq \$3000, %rcx,
%eax	a(%rip),				addq %rcx, %rax leaq 0(,%rax,4),
	%eax, %esi leaq .LC4(%rip),			%rcx	leaq b(%rip), %rax
%rdi	movl \$0,_%eax			%ecx	movl (%rcx,%rax), movl
	call printf@PLT movq				-96(%rbp), %eax add1 \$3, %eax cltq mov1
%rax	-56(%rbp), movq			%esi	-100(%rbp), movslq %esi, %rsi
104(%rbp)	%rax, -			%rsi	%esi, %rsi imulq \$3000, %rsi,
%×mm0	-104(%rbp),				addq %rsi, %rax leaq 0(,%rax,4),
%rdi	.LC5(%rip),			%rsi	leaq c(%rip), %rax
1/	O V I			itoetura v Toenologís	mòvl ''

		1	T	
	\$1, %eax call		%eax	(%rsi,%rax),
	printf@PLT			imull %ecx, %eax
	movl			leal (%rdx,%rax),
	\$0, %eax movq		%ecx	movl
%rdi	-8(%rbp),		%eax	-100(%rbp),
%rdi	xorq		weax	cltq
0/10 01 -	%fs:40,			movl -104(%rbp),
%rdi	je		%edx	movslq %edx, %rdx
	.L20			ımu⊥q
	call		%rdx	\$3000, %rdx,
stack_chk_1	fail@PLT			addq %rdx, %rax
.L20:	leave			leaq′ 0(,%rax,4),
	.cfi_def_cf		%rdx	
a 7, 8	ret			leaq a(%rip), %rax movl
	.cfi_endpro		%rav)	%ecx, (%rdx,
C .LFE6:			%rax)	addl
	.size			\$4, -96(%rbp) addl
main	maın,		.L19:	\$1, -92(%rbp)
	.section			movl -92(%rbp), %eax
	.rodata .align 8			cmpl -88(%rbp), %eax
.LC3:				jl .L20
	. Long 0			movl -88(%rbp), %eax
	.long			sall
	1104006501 .ident			\$2, %eax movl %eax, -96(%rbp)
	"GCC:			jmp .L21
(Ubuntu 7.3.0 7.3.0"	9-16ubuntu3)		.L22:	movl
	.section		%eax	-100(%rbp),
stack,"",@pro	.note.GNU- oabits		, and an	cltq movl
, , , , ,	9		%edx	-104(%rbp),
			, acax	movslq %edx, %rdx
				imulq \$3000, %rdx,
			%rdx	
				addq %rdx, %rax
			%rdx	leaq 0(,%rax,4),
			, and an	leaq a(%rip), %rax
				movi (%rdx,%rax),
			%edx	movl
				-96(%rbp), %eax
				cltq movl
			%ecx	-104(%rbp),
				movslq %ecx, %rcx
			0/	imulq \$3000, %rcx,
			%rcx	addq
				%rcx, %rax leaq 0(,%rax,4),
			%rcx	
				leaq b(%rip), %rax
			0/	movl (%rcx,%rax),
			%ecx	movl
				-96(%rbp), %eax cltg
				movl -100(%rbp),
			%esi	movslq
				%esi, %rsi imulq
			%rsi	\$3000, %rsi,
				addq %rsi, %rax
			<u>.</u> .	leaq 0(,%rax,4),
			%rsi	leaq
				c(%rip), %rax movl
			%eax	(%rsi,%rax),
		Danta Augu	litectura v Tecnologí	

		imull %ecx, %eax leal (%rdx,%rax),
	%ecx	movl
	%eax	-100(%rbp), cltq mov1 -104(%rbp),
	%edx	movslq %edx, %rdx imulg
	%rdx	\$3000, %rdx,
	%rdx	addq %rdx, %rax leaq 0(,%rax,4),
	701 U.X	leaq a(%rip), %rax movl %ecx, (%rdx,
	%rax)	addl
	.L21:	\$1, -96(%rbp) movl -96(%rbp), %eax
	%eax	cmp1 -108(%rbp), jl
	.L18:	.L22 addl \$1, -100(%rbp)
		movl -100(%rbp),
	%eax	cmpl -108(%rbp),
	%eax	jl .L23 addl
	.L17:	\$1, -104(%rbp) movl
	%eax	-104(%rbp),
	%eax	cmpl -108(%rbp),
		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, %rdx movq %rdx, %rax cvtsi2Sdq %rax, %xmm0 movsd LC2(%rip),
	%xmm2	divsd %xmm2, %xmm0
		addsd %xmm1, %xmm0 movsd
	80(%rbp)	%xmm0, - movq
		-48(%rbp), %rdx movq -32(%rbp), %rax subq %rax, %rdx movq %rdx, %rax cvtsi2sdq %rax, %xmm1 movq -40(%rbp), %rdx
		-24(%rbp), %rax subq
16	 toctura v Tocnología	

	%xmm2	%rax, %rdx movq %rdx, %rax cvtsi2sdq %rax, %xmm0 movsd .LC2(%rip),
		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
	ov. d	a(%rip), %rax movl (%rdx,%rax),
	%edx	mov1 a(%rip), %eax mov1 %eax, %esi leaq .LC3(%rip),
	%rdi	movl \$0, %eax call printf@PLT movq -80(%rbp), %rax movq %rax, -
	136(%rbp) %xmm0	movsd -136(%rbp),
	%rdi	leaq .LC4(%rip),
		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	leag .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@ .L26:	
	8	leave .cfi_def_cfa 7,
	.LFE6:	ret .cfi_endproc
		.size main,main .section .rodata .align 8
Dento Argu	itactura v Tacnologí:	a de Computadores

```
.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

# 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;
}</pre>
```

# b) Captura figura1-modificado\_b.c

#### 1.1. TIEMPOS:

.i. iiLivii Oo.				
Modificación	-O0	-01	-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	.file	.file
"figura1-	"figura1-	"figura1-

original.c"		modificado_a.c"		modificado_b.c"	
or iginal.c	.text	mouli icado_a.c	.text .comm	mouli icado_b.c	.text .comm
	.comm s,40000,32		s,40000,32		s,40000,32
	.section .rodata		.section .rodata		.section .rodata
.LC1:	.string	.LC1:	.string	.LC1:	.string
D[MAY 1] - 0/i\n	"R[0]" %i,	D[MAY 4] = 0/i\n	"R[0] = %i,	D[MAY 1] = 0/i\n	"R[0] = %i,
R[MAX-1] = %i\n"  .LC2:		R[MAX-1] = %i\n"  .LC2:		R[MAX-1] = %i\n" .LC2:	
	.string "∖nTiempo		.string "\nTiempo		.string "\nTiempo
(seg.) = %11.9f\r	n" .text	(seg.) = %11.9f\r	ו" .text	(seg.) = %11.9f\r	ı" .text
	.globl		.globl		.globl
	main .type		main .type		main .type
  main:	main, @function	main:	main, @function	  main:	main, @function
.LFB0:	.cfi_startproc	.LFB0:	.cfi startproc	.LFB0:	.cfi_startproc
	pushq		pushq		pushq .
	%rbp .cfi_def_cfa_of		%rbp .cfi_def_cfa_of		%rbp .cfi_def_cfa_of
fset 16	.cfi_offset 6,	fset 16	.cfi_offset 6,	fset 16	.cfi_offset 6,
-16	movq	-16	movq	-16	movq
	%rsp, %rbp_		%rsp, %rbp		%rsp, %rbp_
gister 6	.cfi_def_cfa_re	gister 6	.cfi <u>_def_</u> cfa_re	gister 6	.cfi_def_cfa_re
	subq \$160096, %rsp		subq \$160096, %rsp		subq \$160096, %rsp
	movq %fs:40, %rax		movq %fs:40, %rax		movq %fs:40, %rax
	movq		movq		movq
	%rax, -8(%rbp) xorl		%rax, -8(%rbp) xorl		%rax, -8(%rbp) xorl
	%eax, %eax leaq		%eax, %eax leaq		%eax, %eax leaq
%rax	-160048(%rbp),	%rax	-160048(%rbp),	%rax	-160048(%rbp),
	movq %ray %rsi		movq %ray %rsi		movq %ray %rsi
	%rax, %rsi movl		%rax, %rsi movl		%rax, %rsi movl
	\$0, %edi call		\$0, %edi call		\$0, %edi call
clock_gettime@PL <sup>-</sup>	Т	  clock_gettime@PL1	г	  clock_gettime@PL1	г
Olook_gettime@il	movl	gereimeer E	mov1	gereimeer Er	movl
160072(%rbp)	\$1, -	160072(%rbp)	\$1, -	160072(%rbp)	\$1, -
	jmp .L2		jmp .L2		jmp .L2
.L9:	movl	.L5:	movl	.L5:	movl
160064(%rbn)	\$0, -	160064(%rbp)	\$0, -	160064(%rbp)	\$0, -
160064(%rbp)	movl	100004(WIDH)	movl	700004(%) ph)	movl
160060(%rbp)	\$0, -	160060(%rbp)	\$0, -	160060(%rbp)	\$0, -
	movl \$0, -		movl \$0, -		movl \$0, -
160068(%rbp)	jmp	160068(%rbp)	jmp	160068(%rbp)	jmp
14:	. L3	14:	.L3	14:	.L3
.L4: 	movl	.L4: 	mov1	.L4: 	mov1
%eax	-160068(%rbp),	%eax	-160068(%rbp),	%eax	-160068(%rbp),
	cltq leaq		cltq leaq		cltq leaq
  %rdx	0(,%rax,8),	%rdx	0(,%rax,8),	%rdx	0(,%rax,8),
701 U.A	leaq	701 U.A	leaq	701 UA	leaq
	s(%rip), %rax movl		s(%rip), %rax movl		s(%rip), %rax movl
%eax	(%rdx,%rax),	%eax	(%rdx,%rax),	%eax	(%rdx,%rax),
	leal (%rax,%rax),		leal (%rax,%rax),		leal (%rax,%rax),
%edx	_	%edx	mov1	%edx	mov1
9/004	movl -160072(%rbp),	%00.V	-160072(%rbp),	%00V	-160072(%rbp),
%eax	addl	%eax	addl	%eax	addl
	%edx, %eax addl		%edx, %eax addl		%edx, %eax addl
160064(%rbp)	%eax, -	160064(%rbp)	%eax, -	160064(%rbp)	%eax, -
	addl		movl		movl
160068(%rbp)	\$1, -	%eax	-160068(%rbp),	%eax	-160068(%rbp),
.L3:	cmpl		cltq leaq		cltq leaq
160068(%rbp)	\$4999, -	%rdx	0(,%rax,8),	%rdx	0(,%rax,8),
	jle .L4		leaq		leaq
	mov1		4+s(%rip), %rax movl		4+s(%rip), %rax
160068(%rbp)	\$0, -	%edx	(%rdx,%rax),	%edx	(%rdx,%rax),
	jmp .L5		movl %edx, %eax		movl %edx, %eax
.L6:	movl		%edx, %eax addl %eax, %eax		addl %eax, %eax
%02Y	-160068(%rbp),		addl		addl
%eax	çltq		%edx, %eax subl		%edx, %eax subl
	lead 0(,%rax,8),	%eax	-160072(%rbp),	%eax	-160072(%rbp),
%rdx	(1		addl		addl
		· · · · · · · · · · · · · · · · · · ·	Donto Argu	itectura v Tecnología	o do Computadoros

Medx						
Kedk         (%rdx, %rax)         60068 (%rbp)         \$1, -1         Keax         1-60068 (%rbp)         \$1, -1         Keax         1-60068 (%rbp)         \$1, -1         Keax         1-60068 (%rbp)         \$1, -1         Keax         \$1, -1         Keax         \$1, -1<		leaq 4+s(%rip), %rax	160060(%rbp)	%eax, -	160060(%rbp)	%eax, -
September   Sept	%edx	movl			%eax	
Mean		add⊥		cmpl \$4999, -		
		%eax, %eax	160068(%rbp)	,		leaq
Meax		%edx, %eax		.L4_	%rdx	_
160966(%rbp) add1   160966(%rbp)   1	%02Y	-160072(%rbp),	%oox	-160064(%rbp),		s(%rip), %rax
160868 (%rbp)	//eax	addl	/0Cax	cmpl	%00.Y	
169886 (% rbp)   St.	160060(%rbp)		160060(%rbp)	•	%eax	
.15:				-160060(%rbp),	%edx	_
169686(%rbp)   34999, -	160068(%rbp)  .L5:	_	%eax			
116		cmp⊥ \$4999, -		movl	%eax	
	160068(%rbp)	jle	%eax	-160072(%rbp),		addl'
Meax		.L6_			160064(%rbp)	%eax, -
Seax   190	%eax		160016(%rbn.%rax.	%edx, -		
168672(%rbp)		cmpl -160060(%rbn)		addl	%eax	
160972(%rbp)	%eax			Ψ±/		\$1, %eax
### ### ### ### ### ### ### ### ### ##		. L7 mov1		cmpl		lead
Med	%02Y		160072(%rbp)	,	%rdx	
### ### ### ### ### ##################	/oca^			.L5		4+s(%rip), %rax
160016(%rbp,%rax,   mov1	0/ - 1			-160032(%rbp),	0/1	
169616(%rbp, %rax, %rdx	/weax		%rax	movq .	/weax	
Meax	160016(%rbp,%rax,	%edx, - ,4)		movl		addl
Meax		jmp .L8		\$0, %edi call		%eax, %eax addl
## 1-60072(%rbp),	.L7:	movl	  clock gettime@PL7	Г		%edx, %eax subl
Cltg	%eax	-160072(%rbp),		movq	%eax	-160072(%rbp),
## Wedx			%rdx			
mov1	%edx	-160060(%rbp),	%rax	-160048(%rbp),	160060(%rbp)	
160016(%rbp), %rax, 4)   .1.8:	, acan		70. 47.	subq %rax. %rdx	%eax	-160068(%rbp),
160072(%rbp)	160016(%rbp,%rax,			movq		addl \$2. %eax
1-12:				cvts12sdq		cltq
160072(%rbp)   160072(%rbp)   1610	160072(%rbp)	Ψ±/		movq	%rdy	0(,%rax,8),
1600/2(%rbp)			%rdx		701 UX	leaq
Lead	160072(%rbp)	•	%ray	-160040(%rbp),		WONT
#rax		. L9	701 UX	subq %ray %rdy	%eax	
Movd	9/ray	-160032(%rbp),		movq	%ody	
Sol	761 a.X	movq		cvtsi2sdq	%eux	mov1
Clock_gettime@PLT		mov1'		movsd	%eax	
Clock_gettime@PLT		call	%xmm2			%edx, %eax
## Action	clock_gettime@PLT			%xmm2, %xmm0	100001/0/ : ;	
#rdx	0,	шоvq -160032(%rbp),		%xmm1, %xmm0	160064(%rbp)	
## Subq	%rdx 	movq			%eax	
Subq	%rax	-160048(%rbp),	160056(%rbp)			\$2, %eax
Movd		%rax, %rdx		movI		cltq leag
Cvtsi2sdq		movq %rdx, %rax	%eax	-160016(%rbp),	%rdx	0(,%rax,8),
## A		cvtsi2sdq				leaq 4+s(%rip), %rax
%rdx         movq		movq		leag		WONT
**Tax**  **Judy	%rdx		%rdi	_	%edx	
Subq	%ray	-160040(%rbp),		\$0, %eax		%edx, %eax
Movd   Srdx	, W. U.A.			printf@PLT		%eax, %eax
Cvts12sdq movd wrax, - movd l60088(wrbp)  %xmm2  divsd %xmm2, %xmm0 addsd %xmm1, xxmm0 movsd of xmm1, xxmm0 movsd of xmm1, xxmm0 movsd wxmm1, xxmm0 movsd wxmm0, - movl symm0, - movl symm0, - movl symm0  160056(wrbp)  movl symm0, - movl symm0, - movl symm0, - movl symm0 movsd wxmm0, - movl symm0,		movq	%ray	-160056(%rbp),		%edx, %eax
Movsd   160088(\(\text{rbp}\)   movsd   160088(\(\text{rbp}\)   movsd   160060(\(\text{rbp}\))		cvtsi2sdq	, u.	movq %ray -	%eav	
%xmm2     divsd %xmm2, %xmm0 addsd %xmm1, %xmm0 movsd %xmm0, -     leaq .LC2(%rip), %eax     movl 3, %eax cltq leaq eaq equivalent printf@PLT movl     addl \$3, %eax cltq leaq eq (,%rax,8),		movsd	160088(%rbp)		/UCAA	
160056(%rbp)	%xmm2		0/s/mm0		160060(%rbp)	
### ### ##############################		%xmm2, %xmm0	/0XIIIIIU	leaq	%aay	
160056(%rbp)		%xmm1, %xmm0	%rdi	_	/₀edX	addl
movl printf@PLT 0(,%rax,8), -20(%rbp), %edx movl %rdx	100050(0/			\$1, %eax		c1tq
	100020(%rup)	movl		printf@PLT	0/25 d ) 4	
Donto, Arquitoctura y Tochología do Computadoros	77	-∠⊍(%rbp), %edx				

%eax	movl -160016(%rbp),		<pre>\$0, %eax movq -8(%rbp), %rcx</pre>		leaq s(%rip), %rax movl
	movl %eax, %esi leaq		xorq %fs:40, %rcx je	%eax	(%rdx,%rax),
%rdi	.LC1(%rip), movl		.L7 call	%edx	(%rax,%rax), movl
	<pre>\$0, %eax call printf@PLT</pre>	stack_chk_fail( .L7:	leave	%eax	-160072(%rbp),
%rax	movq -160056(%rbp),	8	.cfi_def_cfa 7,		%edx, %eax addl %eax, -
160088(%rbp)	movq %rax, -	.LFE0:	.cfi_endproc .size	160064(%rbp)	movl -160068(%rbp),
%×mm0	movsd -160088(%rbp),		main,main .section .rodata	%eax	addl \$3, %eax
%rdi	leaq .LC2(%rip),	.LC0:	.align 8 .long		cltq leaq 0(,%rax,8),
	movl \$1, %eax call		0 .long 1104006501	%rdx	leaq 4+s(%rip), %rax
	printf@PLT movl \$0, %eax	7.3.0-16ubuntu3)	.ident "GCC: (Ubuntu 7.3.0"	%edx	movI (%rdx,%rax),
	movq -8(%rbp), %rcx xorq	stack,"",@progbi	.section .note.GNU- ts		movl %edx, %eax addl
	%fs:40, %rcx je .L11				%eax, %eax addl %edx, %eax
stack_chk_fail(	call @PLT			%eax	subl -160072(%rbp),
.L11:	leave .cfi_def_cfa 7,			160060(%rbp)	addl %eax, -
8	ret .cfi_endproc			160068(%rbp)	addl \$4, -
.LFE0:	.size main,main			160069(%rbp)	cmpl \$4999, -
1.00	.section .rodata .align 8			160068(%rbp)	jle .L4
.LC0:	.long 0 .long			%eax	mov1 -160064(%rbp),
	1104006501 .ident "GCC: (Ubuntu			160060(%rbp)	cmpl %eax, - cmovle
7.3.0-16ubuntu3)	7.3.0" .section .note.GNU-			%eax	-160060(%rbp),
stack,"",@progbi	ts				%eax, %edx mov1 -160072(%rbp),
				%eax	cltq movl
				160016(%rbp,%rax,	%edx, - 4) addl \$1, -
				160072(%rbp)  .L2:	cmpl \$4000, -
				160072(%rbp)	\$40000, - jle ;L5
					.L5 leaq -160032(%rbp),
				%rax	movq %rax, %rsi
					movl \$0, %edi call
				clock_gettime@PL1	movq
				%rdx	-160032(%rbp), 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
			Donto Augus	<u></u>	cvtsi2sdq %rax, %xmm0 movsd a de Computadores

```
.LC0(%rip),
%xmm2
                             havib
                                        %xmm0
                             addsd %xmm1, %xmm0
                             movsď
%xmm0,
160056(%rbp)
                             mov1
-20(%rbp), %edx
mov1
-160016(%rbp),
%eax
                             movl
%eax, %esi
                             leaq'
.LC1(%rip),
%rdi
                             movl
$0, %eax
call
printf@PLT
                             movq
-160056(%rbp),
%rax
160088(%rbp)
                             movsd
-160088(%rbp),
%xmm0
                             leaq
.LC2(%rip),
%rdi
                            movl
$1, %eax
call
printf@PLT
movl
$0, %eax
movg
-8(%rbp), %rcx
                             xorq
%fs:40, %rcx
                             je
.L7
call
__stack_chk_fail@PLT
.L7:
                             leave
.cfi_def_cfa 7,
                             ret
.cfi endproc
.LFE0:
                             .size
main, .-main
.section
.rodata
.align 8
.LC0:
                             .long
                             .long
1104006501
ident
ident
"GCC: (Ubuntu
7.3.0-16ubuntu3) 7.3.0"
.section
. 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 (*Double precision- real Alpha X Plus Y*) que multiplica un vector por una constante y los suma a otro vector (Lección 3/Tema 1):

for 
$$(i=1; i \le N, i++)$$
  $y[i] = a*x[i] + y[i];$ 

- 2.1. Genere los programas en ensamblador para cada una de las siguientes opciones de optimización del compilador: -O0, -Os, -O2, -O3. Explique las diferencias que se observan en el código justificando al mismo tiempo las mejoras en velocidad que acarrean. Incorpore los códigos al cuaderno de prácticas y destaque las diferencias entre ellos.
- 2.2. (Ejercicio EXTRA) Para la mejor de las opciones, obtenga los tiempos de ejecución con distintos valores de N y determine para su sistema los valores de Rmax (valor máximo del número de operaciones en coma flotante por unidad de tiempo), Nmax (valor de N para el que se consigue Rmax), y N1/2 (valor de N para el que se obtiene Rmax/2). Estime el

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

CAPTURA CÓDIGO FUENTE: daxpy.c

	-O0	-Os	-02	-O3
	0.785696	0.320731	0.320363	0.309885
Tiempos ejec.	029	439	732	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