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):
Grupo de prácticas:
Fecha de entrega:
Fecha evaluación en clase:

Denominación de marca del chip de procesamiento o procesador (se encuentra en /proc/cpuinfo): *Intel(R) Core(TM) i7-2670QM CPU @ 2.20GHz*

Sistema operativo utilizado: *Ubuntu 18.04 x64*

Versión de gcc utilizada: *gcc version 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

```
mar 22 may - 18:02  ~/Escritorio/ETSIIT/2º Curso/2º Cuatrimestre/AC/Practicas/Practica_4
@david  lscpu
Arquitectura:                    x86_64
modo(s) de operación de las CPUs:    32-bit, 64-bit
Orden de los bytes:              Little Endian
CPU(s):                          8
Lista de la(s) CPU(s) en línea:      0-7
Hilo(s) de procesamiento por núcleo: 2
Núcleo(s) por «socket»:              4
«Socket(s)»                      1
Modo(s) NUMA:                    1
ID de fabricante:                GenuineIntel
Familia de CPU:                  6
Modelo:                          42
Nombre del modelo:               Intel(R) Core(TM) i7-2670QM CPU @ 2.20GHz
Revisión:                        7
CPU MHz:                         979.124
CPU MHz máx.:                    3100,0000
CPU MHz mín.:                    800,0000
BogoMIPS:                        4390.30
Virtualización:                  VT-x
Caché L1d:                       32K
Caché L1i:                       32K
Caché L2:                        256K
Caché L3:                        6144K
CPU(s) del nodo NUMA 0:              0-7
Indicadores:                     fpu vme de pse tsc msr pae mce cx8 apic sep mtrr pge mca cmov pat pse36 clflush dts acpi mmx fxsr sse sse2 ss ht t
m pbe syscall nx rdtscp lm constant_tsc arch_perfmon pebs bts rep_good nopl xtopology nonstop_tsc cpuid aperfmperf pni pclmulqdq dtes64 monitor ds_cpl
vmx est tm2 ssse3 cx16 xtpr pdcm pcid sse4_1 sse4_2 x2apic popcnt tsc_deadline_timer aes xsave avx lahf_lm epb pti tpr_shadow vnmi flexpriority ept vpi
d xsaveopt dtherm ida arat pln pts
```

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

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

int main(int argc, char const *argv[]) {
  if (argc < 2) {
    fprintf(stderr, "ERROR: falta numero de filas y columnas\n");
    exit(1);
  }

  unsigned n, i, j, k;
  n = strtol(argv[1], NULL, 10);

  int **a, **b, **c;
  a = (int **)malloc(n * sizeof(int *));
  b = (int **)malloc(n * sizeof(int *));
  c = (int **)malloc(n * sizeof(int *));
  for (i = 0; i < n; i++) {
    a[i] = (int *)malloc(n * sizeof(int));
    b[i] = (int *)malloc(n * sizeof(int));
    c[i] = (int *)malloc(n * sizeof(int));
  }

  // Inicializcion
  for (i = 0; i < n; i++) {
    for (j = 0; j < n; j++) {
      a[i][j] = 0;
      b[i][j] = 1;
      c[i][j] = 2;
    }
  }

  struct timespec cgt1, cgt2;
  double ncgt;

  // Multiplicacion
  clock_gettime(CLOCK_REALTIME, &cgt1);
  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));

  if (n < 15) {
    printf("M1:\n");
    printMatriz(n, b);
    printf("M2:\n");
    printMatriz(n, c);
    printf("Sol:\n");
    printMatriz(n, a);
  } else
    printf("Tiempo = %11.9f\t Primera = %d\t Ultima=%d\n", ncgt, a[0]
[0],
           a[n - 1][n - 1]);

  return 0;
}
```

1.1. MODIFICACIONES REALIZADAS (al menos dos modificaciones):

Modificación a) –explicación-: Convertir la matriz en un vector para hacer alineamiento de memoria

Modificación b) –explicación-: Cambiar los bucles j y k para optimizar el acceso a memoria de la multiplicación

1.1. CÓDIGOS FUENTE MODIFICACIONES

a) Captura de `pmm-secuencial-modificado_a.c`

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

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

void pmm(int *a, int *b, int *c, unsigned n, struct timespec *cgt1,
         struct timespec *cgt2) {
  unsigned i, j, k;

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

  clock_gettime(CLOCK_REALTIME, cgt2);
}

int main(int argc, char const *argv[]) {
  if (argc < 2) {
    fprintf(stderr, "ERROR: falta numero de filas y columnas\n");
    exit(1);
  }

  unsigned n = strtol(argv[1], NULL, 10);
  unsigned i, j;
  int *a, *b, *c, *new_a, *new_b, *new_c;
  unsigned n_cuadrado = n * n;
  a = (int *)malloc(sizeof(int) * n_cuadrado + 63);
  new_a = (int *)(((long int)a + 63) & ~(63));
  b = (int *)malloc(sizeof(int) * n_cuadrado + 63);
  new_b = (int *)(((long int)b + 63) & ~(63));
  c = (int *)malloc(sizeof(int) * n_cuadrado + 63);
  new_c = (int *)(((long int)c + 63) & ~(63));

  // Inicializcion
  for (i = 0; i < n; i++) {
    for (j = 0; j < n; j++) {
      new_a[(i * n) + j] = 0;
      new_b[(i * n) + j] = 2;
      new_c[(i * n) + j] = 1;
    }
  }

  struct timespec cgt1, cgt2;
  double ncgt;

  pmm(new_a, new_b, new_c, n, &cgt1, &cgt2);

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

  if (n < 15) {
    printf("M1:\n");
    printMatriz(n, new_b);
    printf("M2:\n");
    printMatriz(n, new_c);
    printf("Sol:\n");
    printMatriz(n, new_a);
  } else
    printf("Tiempo = %11.9f\t Primera = %d\t Ultima=%d\n", ncgt, new_a[0],
           new_a[(n - 1) + (n - 1)]);
```

```
  free(a);
  free(b);
  free(c);

  return 0;
}
```

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

```
mié 30 may - 19:08  ~/Escritorio/ETSIIT/2º Curso/2º Cuatrimestre/AC/Practicas/Practica_4
@david  gcc -O2 -lrt pmm-secuencial-modificado_a.c -o pmm-secuencial-modificado_a

mié 30 may - 19:08  ~/Escritorio/ETSIIT/2º Curso/2º Cuatrimestre/AC/Practicas/Practica_4
@david  ./pmm-secuencial-modificado_a 5
M1:
1 1 1 1 1
1 1 1 1 1
1 1 1 1 1
1 1 1 1 1
1 1 1 1 1
M2:
2 2 2 2 2
2 2 2 2 2
2 2 2 2 2
2 2 2 2 2
2 2 2 2 2
Sol:
10 10 10 10 10
10 10 10 10 10
10 10 10 10 10
10 10 10 10 10
10 10 10 10 10
```

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

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

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

int main(int argc, char const *argv[]) {
  if (argc < 2) {
    fprintf(stderr, "ERROR: falta numero de filas y columnas\n");
    exit(1);
  }

  unsigned n, i, j, k;
  n = strtol(argv[1], NULL, 10);

  int **a, **b, **c;
  a = (int **)malloc(n * sizeof(int *));
  b = (int **)malloc(n * sizeof(int *));
  c = (int **)malloc(n * sizeof(int *));
  for (i = 0; i < n; i++) {
    a[i] = (int *)malloc(n * sizeof(int));
    b[i] = (int *)malloc(n * sizeof(int));
    c[i] = (int *)malloc(n * sizeof(int));
  }

  // Inicializcion

  for (i = 0; i < n; i++) {
    for (j = 0; j < n; j++) {
      a[i][j] = 0;
      b[i][j] = 1;
      c[i][j] = 2;
    }
```

```
  }

  struct timespec cgt1, cgt2;
  double ncgt;

  // Multiplicacion
  clock_gettime(CLOCK_REALTIME, &cgt1);
  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));

  if (n < 15) {
    printf("M1:\n");
    printMatriz(n, b);
    printf("M2:\n");
    printMatriz(n, c);
    printf("Sol:\n");
    printMatriz(n, a);
  } else {
    printf("Tiempo = %11.9f\t Primera = %d\t Ultima=%d\n", ncgt, a[0]
[0],
           a[n - 1][n - 1]);
  }

  // Eliminar memoria
  for (i = 0; i < n; i++) {
    free(a[i]);
    free(b[i]);
    free(c[i]);
  }

  free(a);
  free(b);
  free(c);

  return 0;
}
```

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

```
mié 30 may - 19:10  ~/Escritorio/ETSIIT/2º Curso/2º Cuatrimestre/AC/Practicas/Practica_4
@david  gcc -O2 -lrt pmm-secuencial-modificado_b.c -o pmm-secuencial-modificado_b

mié 30 may - 19:13  ~/Escritorio/ETSIIT/2º Curso/2º Cuatrimestre/AC/Practicas/Practica_4
@david  ./pmm-secuencial-modificado_b 5
M1:
1 1 1 1 1
1 1 1 1 1
1 1 1 1 1
1 1 1 1 1
1 1 1 1 1
M2:
2 2 2 2 2
2 2 2 2 2
2 2 2 2 2
2 2 2 2 2
2 2 2 2 2
Sol:
10 10 10 10 10
10 10 10 10 10
10 10 10 10 10
10 10 10 10 10
10 10 10 10 10
```

1.1. TIEMPOS:

Tamaño: 1000

Modificación	**-O2**
Sin modificar	9,084460365
Modificación a)	1,053379518
Modificación b)	8,653854064

1.1. COMENTARIOS SOBRE LOS RESULTADOS:

Se ve una gran mejora en la modificacion a ya que al combertir la matriz a un vector se requieren muchos menos accesos a memoria que son lo que realmente produce que el tiempo de ejecución aumente.

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
                .section        .rodata.str1.1,"aMS",@progbits,1
.LC0:
                .string "%d "
                .text
                .p2align 4,,15
                .globl  printMatriz
                .type   printMatriz, @function
printMatriz:
.LFB52:
                .cfi_startproc
                testl   %edi, %edi
                jle     .L7
                leal    -1(%rdi), %eax
                pushq   %r14
                .cfi_def_cfa_offset 16
                .cfi_offset 14, -16
                pushq   %r13
                .cfi_def_cfa_offset 24
                .cfi_offset 13, -24
                pushq   %r12
                .cfi_def_cfa_offset 32
                .cfi_offset 12, -32
                pushq   %rbp
                .cfi_def_cfa_offset 40
                .cfi_offset 6, -40
                leaq    .LC0(%rip), %rbp
                pushq   %rbx
                .cfi_def_cfa_offset 48
                .cfi_offset 3, -48
                leaq    8(%rsi,%rax,8), %r13
                leaq    4(,%rax,4), %r12
                movq    %rsi, %r14
                .p2align 4,,10
                .p2align 3
.L3:
                xorl
```

pmm-secuencial-modificado_b.s

```
                .file   "pmm-secuencial-modificado_a.c"
                .text
                .section        .rodata.str1.1,"aMS",@progbits,1
.LC0:
                .string "%d "
                .text
                .p2align 4,,15
                .globl  printMatriz
                .type   printMatriz, @function
printMatriz:
.LFB52:
                .cfi_startproc
                testl   %edi, %edi
                jle     .L7
                leal    -1(%rdi), %eax
                pushq   %r15
                .cfi_def_cfa_offset 16
                .cfi_offset 15, -16
                pushq   %r14
                .cfi_def_cfa_offset 24
                .cfi_offset 14, -24
                pushq   %r13
                .cfi_def_cfa_offset 32
                .cfi_offset 13, -32
                pushq   %r12
                .cfi_def_cfa_offset 40
                .cfi_offset 12, -40
                leaq    .LC0(%rip), %r13
                pushq   %rbp
                .cfi_def_cfa_offset 48
                .cfi_offset 6, -48
                salq    $2, %rax
                pushq   %rbx
                .cfi_def_cfa_offset 56
                .cfi_offset 3, -56
                leaq    4(%rsi,%rax), %r15
                leaq
```

pmm-secuencial-modificado_c.s

```
                .file   "pmm-secuencial-modificado_b.c"
                .text
                .section        .rodata.str1.1,"aMS",@progbits,1
.LC0:
                .string "%d "
                .text
                .p2align 4,,15
                .globl  printMatriz
                .type   printMatriz, @function
printMatriz:
.LFB52:
                .cfi_startproc
                testl   %edi, %edi
                jle     .L7
                leal    -1(%rdi), %eax
                pushq   %r14
                .cfi_def_cfa_offset 16
                .cfi_offset 14, -16
                pushq   %r13
                .cfi_def_cfa_offset 24
                .cfi_offset 13, -24
                pushq   %r12
                .cfi_def_cfa_offset 32
                .cfi_offset 12, -32
                pushq   %rbp
                .cfi_def_cfa_offset 40
                .cfi_offset 6, -40
                leaq    .LC0(%rip), %rbp
                pushq   %rbx
                .cfi_def_cfa_offset 48
                .cfi_offset 3, -48
                leaq    8(%rsi,%rax,8), %r13
                leaq    4(,%rax,4), %r12
                movq    %rsi, %r14
                .p2align 4,,10
                .p2align 3
.L3:
                xorl
```

```
                %ebx, %ebx
                .p2align 4,,10
                .p2align 3
.L4:
                movq
                (%r14), %rax
                movq
                %rbp, %rsi
                movl
                $1, %edi
                movl
                (%rax,%rbx),
%edx
                xorl
                %eax, %eax
                addq
                $4, %rbx
                call
__printf_chk@PLT
                cmpq
                %rbx, %r12
                jne
                .L4
                movl
                $10, %edi
                addq
                $8, %r14
                call
                putchar@PLT
                cmpq
                %r13, %r14
                jne
                .L3
                popq
                %rbx
                .cfi_def_cfa_of
fset 40
                popq
                %rbp
                .cfi_def_cfa_of
fset 32
                popq
                %r12
                .cfi_def_cfa_of
fset 24
                popq
                %r13
                .cfi_def_cfa_of
fset 16
                popq
                %r14
                .cfi_def_cfa_of
fset 8
                ret
.L7:
                .cfi_restore 3
                .cfi_restore 6
                .cfi_restore 12
                .cfi_restore 13
                .cfi_restore 14
                rep ret
                .cfi_endproc
.LFE52:
                .size
                printMatriz, .-
printMatriz
                .section
                .rodata.str1.8,
"aMS",@progbits,1
                .align 8
.LC1:
                .string
                "ERROR: falta
numero de filas y columnas\n"
                .section
                .rodata.str1.1
.LC2:
                .string
                "M1:"
.LC3:
                .string
                "M2:"
.LC4:
                .string
                "Sol:"
                .section
                .rodata.str1.8
                .align 8
.LC6:
                .string
                "Tiempo =
%11.9f\t Primera = %d\t Ultima=
%d\n"
                .section
                .text.startup,"
ax",@progbits
                .p2align 4,,15
                .globl
                main
                .type
                main, @function
main:
.LFB53:
                .cfi_startproc
                pushq
                %r15
                .cfi_def_cfa_of
fset 16
                .cfi_offset 15,
-16
                pushq
                %r14
                4(%rax), %r14
                movq
                %rsi, %r12
                subq
                $8, %rsp
                .cfi_def_cfa_of
fset 64
                .p2align 4,,10
                .p2align 3
.L3:
                leaq
                (%r14,%r12),
%rbp
                movq
                %r12, %rbx
                .p2align 4,,10
                .p2align 3
.L4:
                movl
                (%rbx), %edx
                xorl
                %eax, %eax
                movq
                %r13, %rsi
                movl
                $1, %edi
                addq
                $4, %rbx
                call
__printf_chk@PLT
                cmpq
                %rbp, %rbx
                jne
                .L4
                movl
                $10, %edi
                addq
                $4, %r12
                call
                putchar@PLT
                cmpq
                %r15, %r12
                jne
                .L3
                addq
                $8, %rsp
                .cfi_def_cfa_of
fset 56
                popq
                %rbx
                .cfi_def_cfa_of
fset 48
                popq
                %rbp
                .cfi_def_cfa_of
fset 40
                popq
                %r12
                .cfi_def_cfa_of
fset 32
                popq
                %r13
                .cfi_def_cfa_of
fset 24
                popq
                %r14
                .cfi_def_cfa_of
fset 16
                popq
                %r15
                .cfi_def_cfa_of
fset 8
                ret
.L7:
                .cfi_restore 3
                .cfi_restore 6
                .cfi_restore 12
                .cfi_restore 13
                .cfi_restore 14
                .cfi_restore 15
                rep ret
                .cfi_endproc
.LFE52:
                .size
                printMatriz, .-
printMatriz
                .p2align 4,,15
                .globl
                pmm
                .type
                pmm, @function
pmm:
.LFB53:
                .cfi_startproc
                pushq
                %r15
                .cfi_def_cfa_of
fset 16
                .cfi_offset 15,
-16
                pushq
                %r14
                .cfi_def_cfa_of
fset 24
                .cfi_offset 14,
-24
                movl
                %ecx, %r14d
                pushq
                %r13
                .cfi_def_cfa_of
fset 32
                %ebx, %ebx
                .p2align 4,,10
                .p2align 3
.L4:
                movq
                (%r14), %rax
                movq
                %rbp, %rsi
                movl
                $1, %edi
                movl
                (%rax,%rbx),
%edx
                xorl
                %eax, %eax
                addq
                $4, %rbx
                call
__printf_chk@PLT
                cmpq
                %rbx, %r12
                jne
                .L4
                movl
                $10, %edi
                addq
                $8, %r14
                call
                putchar@PLT
                cmpq
                %r13, %r14
                jne
                .L3
                popq
                %rbx
                .cfi_def_cfa_of
fset 40
                popq
                %rbp
                .cfi_def_cfa_of
fset 32
                popq
                %r12
                .cfi_def_cfa_of
fset 24
                popq
                %r13
                .cfi_def_cfa_of
fset 16
                popq
                %r14
                .cfi_def_cfa_of
fset 8
                ret
.L7:
                .cfi_restore 3
                .cfi_restore 6
                .cfi_restore 12
                .cfi_restore 13
                .cfi_restore 14
                rep ret
                .cfi_endproc
.LFE52:
                .size
                printMatriz, .-
printMatriz
                .section
                .rodata.str1.8,
"aMS",@progbits,1
                .align 8
.LC1:
                .string
                "ERROR: falta
numero de filas y columnas\n"
                .section
                .rodata.str1.1
.LC2:
                .string
                "M1:"
.LC3:
                .string
                "M2:"
.LC4:
                .string
                "Sol:"
                .section
                .rodata.str1.8
                .align 8
.LC6:
                .string
                "Tiempo =
%11.9f\t Primera = %d\t Ultima=
%d\n"
                .section
                .text.startup,"
ax",@progbits
                .p2align 4,,15
                .globl
                main
                .type
                main, @function
main:
.LFB53:
                .cfi_startproc
                pushq
                %r15
                .cfi_def_cfa_of
fset 16
                .cfi_offset 15,
-16
                pushq
                %r14
```

```
                .cfi_def_cfa_of
fset 24
                .cfi_offset 14,
-24
                pushq
                %r13
                .cfi_def_cfa_of
fset 32
                .cfi_offset 13,
-32
                pushq
                %r12
                .cfi_def_cfa_of
fset 40
                .cfi_offset 12,
-40
                pushq
                %rbp
                .cfi_def_cfa_of
fset 48
                .cfi_offset 6,
-48
                pushq
                %rbx
                .cfi_def_cfa_of
fset 56
                .cfi_offset 3,
-56
                subq
                $88, %rsp
                .cfi_def_cfa_of
fset 144
                movq
                %fs:40, %rax
                movq
                %rax, 72(%rsp)
                xorl
                %eax, %eax
                cmpl
                $1, %edi
                jle
                .L34
                movq
                8(%rsi), %rdi
                movl
                $10, %edx
                xorl
                %esi, %esi
                call
                strtol@PLT
                movl
                %eax, %r14d
                movq
                %rax, %r15
                movq
                %rax, 8(%rsp)
                leaq
                0(,%r14,8),
%r12
                movq
                %r12, %rdi
                call
                malloc@PLT
                movq
                %r12, %rdi
                movq
                %rax, %rbx
                call
                malloc@PLT
                movq
                %r12, %rdi
                movq
                %rax, %rbp
                call
                malloc@PLT
                testl
                %r15d, %r15d
                movq
                %rax, %r12
                je
                .L13
                movq
                %r15, %rax
                salq
                $2, %r14
                xorl
                %r15d, %r15d
                subl
                $1, %eax
                leaq
                1(%rax), %r13
                movq
                %rax, 16(%rsp)
                leaq
                0(,%r13,8),
%rax
                movq
                %r13, 24(%rsp)
                movq
                %rax, %r13
                .p2align 4,,10
                .p2align 3
.L14:
                movq
                %r14, %rdi
                call
                malloc@PLT
                movq
                %r14, %rdi
                movq
                %rax, (%rbx,
%r15)
                call
                .cfi_offset 13,
-32
                pushq
                %r12
                .cfi_def_cfa_of
fset 40
                .cfi_offset 12,
-40
                movq
                %rsi, %r12
                pushq
                %rbp
                .cfi_def_cfa_of
fset 48
                .cfi_offset 6,
-48
                pushq
                %rbx
                .cfi_def_cfa_of
fset 56
                .cfi_offset 3,
-56
                movq
                %r8, %rsi
                movq
                %rdi, %rbx
                xorl
                %edi, %edi
                movq
                %rdx, %r13
                subq
                $8, %rsp
                .cfi_def_cfa_of
fset 64
                movq
                %r9, %rbp
                call

clock_gettime@PLT
                testl
                %r14d, %r14d
                je
                .L13
                movl
                %r14d, %r10d
                xorl
                %r15d, %r15d
                xorl
                %r11d, %r11d
                .p2align 4,,10
                .p2align 3
.L12:
                movl
                %r10d, %r9d
                movl
                %r15d, %r8d
                subl
                %r14d, %r9d
                movl
                %r9d, %ecx
                .p2align 4,,10
                .p2align 3
.L16:
                movl
                %ecx, %eax
                leaq
                (%r12,%rax,4),
%rdx
                movl
                %r9d, %eax
                .p2align 4,,10
                .p2align 3
.L14:
                leal
                (%rax,%r8),
%esi
                movl
                %eax, %edi
                addl
                $1, %eax
                movl
                0(%r13,%rsi,4),
%esi
                imull
                (%rdx), %esi
                addl
                %esi, (%rbx,
%rdi,4)
                cmpl
                %eax, %r10d
                jne
                .L14
                addl
                $1, %ecx
                addl
                %r14d, %r8d
                cmpl
                %r10d, %ecx
                jne
                .L16
                addl
                $1, %r11d
                addl
                %r14d, %r10d
                subl
                %r14d, %r15d
                cmpl
                %r11d, %r14d
                jne
                .L12
.L13:
                addq
                $8, %rsp
                .cfi_def_cfa_of
fset 24
                .cfi_offset 14,
-24
                pushq
                %r13
                .cfi_def_cfa_of
fset 32
                .cfi_offset 13,
-32
                pushq
                %r12
                .cfi_def_cfa_of
fset 40
                .cfi_offset 12,
-40
                pushq
                %rbp
                .cfi_def_cfa_of
fset 48
                .cfi_offset 6,
-48
                pushq
                %rbx
                .cfi_def_cfa_of
fset 56
                .cfi_offset 3,
-56
                subq
                $88, %rsp
                .cfi_def_cfa_of
fset 144
                movq
                %fs:40, %rax
                movq
                %rax, 72(%rsp)
                xorl
                %eax, %eax
                cmpl
                $1, %edi
                jle
                .L38
                movq
                8(%rsi), %rdi
                movl
                $10, %edx
                xorl
                %esi, %esi
                call
                strtol@PLT
                movl
                %eax, %r14d
                movq
                %rax, %r15
                movq
                %rax, (%rsp)
                leaq
                0(,%r14,8),
%r12
                movq
                %r12, %rdi
                call
                malloc@PLT
                movq
                %r12, %rdi
                movq
                %rax, %rbx
                call
                malloc@PLT
                movq
                %r12, %rdi
                movq
                %rax, %rbp
                call
                malloc@PLT
                testl
                %r15d, %r15d
                movq
                %rax, %r12
                je
                .L13
                movl
                %r15d, %eax
                salq
                $2, %r14
                xorl
                %r15d, %r15d
                subl
                $1, %eax
                leaq
                1(%rax), %r13
                movl
                %eax, 20(%rsp)
                movq
                %rax, 8(%rsp)
                leaq
                0(,%r13,8),
%rax
                movq
                %r13, 24(%rsp)
                movq
                %rax, %r13
                .p2align 4,,10
                .p2align 3
.L14:
                movq
                %r14, %rdi
                call
                malloc@PLT
                movq
                %r14, %rdi
                movq
                %rax, (%rbx,
```

```
                malloc@PLT
                movq
                %r14, %rdi
                movq
                %rax, 0(%rbp,
%r15)
                call
                malloc@PLT
                movq
                %rax,
(%r12,%r15)
                addq
                $8, %r15
                cmpq
                %r15, %r13
                jne
                .L14
                movq
                24(%rsp), %r13
                xorl
                %r14d, %r14d
                leaq
                0(,%r13,4), %r8
                .p2align 4,,10
                .p2align 3
.L16:
                movq
                (%rbx,%r14),
%rsi
                movq
                0(%rbp,%r14),
%rcx
                xorl
                %eax, %eax
                movq
                (%r12,%r14),
%rdx
                .p2align 4,,10
                .p2align 3
.L15:
                movl
                $0, (%rsi,%rax)
                movl
                $1, (%rcx,%rax)
                movl
                $2, (%rdx,%rax)
                addq
                $4, %rax
                cmpq
                %rax, %r8
                jne
                .L15
                addq
                $8, %r14
                cmpq
                %r15, %r14
                jne
                .L16
                leaq
                32(%rsp), %rsi
                xorl
                %edi, %edi
                call
clock_gettime@PLT
                movq
                16(%rsp), %rax
                xorl
                %r11d, %r11d
                leaq
                4(,%rax,4),
%r15
                .p2align 4,,10
                .p2align 3
.L25:
                movq
                (%rbx,%r11),
%r9
                movq
                0(%rbp,%r11),
%r8
                leaq
                (%r15,%r9),
%r10
                movq
                %r9, %rsi
                .p2align 4,,10
                .p2align 3
.L22:
                movl
                (%rsi), %ecx
                movq
                %rsi, %rdi
                xorl
                %eax, %eax
                subq
                %r9, %rdi
                .p2align 4,,10
                .p2align 3
.L19:
                movq
                (%r12,%rax,8),
%rdx
                movl
                (%rdx,%rdi),
%edx
                imull
                (%r8,%rax,4),
%edx
                addq
                $1, %rax
                addl
                %edx, %ecx
                .cfi_def_cfa_of
fset 56
                movq
                %rbp, %rsi
                xorl
                %edi, %edi
                popq
                %rbx
                .cfi_def_cfa_of
fset 48
                popq
                %rbp
                .cfi_def_cfa_of
fset 40
                popq
                %r12
                .cfi_def_cfa_of
fset 32
                popq
                %r13
                .cfi_def_cfa_of
fset 24
                popq
                %r14
                .cfi_def_cfa_of
fset 16
                popq
                %r15
                .cfi_def_cfa_of
fset 8
                jmp
clock_gettime@PLT
                .cfi_endproc
.LFE53:
                .size
                pmm, .-pmm
                .section
                .rodata.str1.8,
"aMS",@progbits,1
                .align 8
.LC1:
                .string
                "ERROR: falta
numero de filas y columnas\n"
                .section
                .rodata.str1.1
.LC2:
                .string
                "M1:"
.LC3:
                .string
                "M2:"
.LC4:
                .string
                "Sol:"
                .section
                .rodata.str1.8
                .align 8
.LC6:
                .string
                "Tiempo =
%11.9f\t Primera = %d\t Ultima=
%d\n"
                .section
                .text.startup,"
ax",@progbits
                .p2align 4,,15
                .globl
                main
                .type
                main, @function
main:
.LFB54:
                .cfi_startproc
                pushq
                %r15
                .cfi_def_cfa_of
fset 16
                .cfi_offset 15,
-16
                pushq
                %r14
                .cfi_def_cfa_of
fset 24
                .cfi_offset 14,
-24
                pushq
                %r13
                .cfi_def_cfa_of
fset 32
                .cfi_offset 13,
-32
                pushq
                %r12
                .cfi_def_cfa_of
fset 40
                .cfi_offset 12,
-40
                pushq
                %rbp
                .cfi_def_cfa_of
fset 48
                .cfi_offset 6,
-48
                pushq
                %rbx
                .cfi_def_cfa_of
fset 56
                .cfi_offset 3,
-56
                subq
                $72, %rsp
%r15)
                call
                malloc@PLT
                movq
                %r14, %rdi
                movq
                %rax, 0(%rbp,
%r15)
                call
                malloc@PLT
                movq
                %rax,
(%r12,%r15)
                addq
                $8, %r15
                cmpq
                %r15, %r13
                jne
                .L14
                movq
                24(%rsp), %r13
                xorl
                %r14d, %r14d
                leaq
                0(,%r13,4), %r8
                .p2align 4,,10
                .p2align 3
.L16:
                movq
                (%rbx,%r14),
%rsi
                movq
                0(%rbp,%r14),
%rcx
                xorl
                %eax, %eax
                movq
                (%r12,%r14),
%rdx
                .p2align 4,,10
                .p2align 3
.L15:
                movl
                $0, (%rsi,%rax)
                movl
                $1, (%rcx,%rax)
                movl
                $2, (%rdx,%rax)
                addq
                $4, %rax
                cmpq
                %rax, %r8
                jne
                .L15
                addq
                $8, %r14
                cmpq
                %r14, %r15
                jne
                .L16
                leaq
                32(%rsp), %rsi
                xorl
                %edi, %edi
                call
clock_gettime@PLT
                movq
                8(%rsp), %rax
                xorl
                %r11d, %r11d
                leaq
                4(,%rax,4),
%r15
                .p2align 4,,10
                .p2align 3
.L27:
                movq
                (%rbx,%r11),
%r9
                movq
                0(%rbp,%r11),
%r8
                leaq
                (%r15,%r9),
%r10
                movq
                %r9, %rsi
                .p2align 4,,10
                .p2align 3
.L22:
                movl
                (%rsi), %ecx
                movq
                %rsi, %rdi
                xorl
                %eax, %eax
                subq
                %r9, %rdi
                .p2align 4,,10
                .p2align 3
.L19:
                movq
                (%r12,%rax,8),
%rdx
                movl
                (%rdx,%rdi),
%edx
                imull
                (%r8,%rax,4),
%edx
                addq
                $1, %rax
```

```
                cmpq
                %r13, %rax
                movl
                %ecx, (%rsi)
                jne
                .L19
                addq
                $4, %rsi
                cmpq
                %rsi, %r10
                jne
                .L22
                addq
                $8, %r11
                cmpq
                %r11, %r14
                jne
                .L25
                leaq
                48(%rsp), %rsi
                xorl
                %edi, %edi
                call
clock_gettime@PLT
                cmpl
                $14, 8(%rsp)
                jbe
                .L18
                movq
                16(%rsp), %rdi
                pxor
                %xmm0, %xmm0
                pxor
                %xmm1, %xmm1
                movq
                (%rbx), %rdx
                leaq
                .LC6(%rip),
%rsi
                movq
                (%rbx,%rdi,8),
%rax
                movl
                (%rdx), %edx
                movl
                (%rax,%rdi,4),
%ecx
                movq
                56(%rsp), %rax
                movl
                $1, %edi
                subq
                40(%rsp), %rax
                cvtsi2sdq
                %rax, %xmm0
                movq
                48(%rsp), %rax
                subq
                32(%rsp), %rax
                cvtsi2sdq
                %rax, %xmm1
                movl
                $1, %eax
                divsd
                .LC5(%rip),
%xmm0
                addsd
                %xmm1, %xmm0
                call
__printf_chk@PLT
.L24:
                xorl
                %eax, %eax
                movq
                72(%rsp), %rbx
                xorq
                %fs:40, %rbx
                jne
                .L35
                addq
                $88, %rsp
                .cfi_remember_s
tate
                .cfi_def_cfa_of
fset 56
                popq
                %rbx
                .cfi_def_cfa_of
fset 48
                popq
                %rbp
                .cfi_def_cfa_of
fset 40
                popq
                %r12
                .cfi_def_cfa_of
fset 32
                popq
                %r13
                .cfi_def_cfa_of
fset 24
                popq
                %r14
                .cfi_def_cfa_of
fset 16
                popq
                %r15
                .cfi_def_cfa_of
fset 8
                ret
.L13:
                .cfi_def_cfa_of
fset 128
                movq
                %fs:40, %rax
                movq
                %rax, 56(%rsp)
                xorl
                %eax, %eax
                cmpl
                $1, %edi
                jle
                .L33
                movq
                8(%rsi), %rdi
                movl
                $10, %edx
                xorl
                %esi, %esi
                call
                strtol@PLT
                movq
                %rax, %rbp
                movl
                %eax, (%rsp)
                imull
                %eax, %eax
                leaq
                63(,%rax,4),
%r14
                movq
                %r14, %rdi
                call
                malloc@PLT
                movq
                %r14, %rdi
                leaq
                63(%rax), %rbx
                movq
                %rax, %r12
                call
                malloc@PLT
                movq
                %r14, %rdi
                leaq
                63(%rax), %r15
                movq
                %rax, %r13
                call
                malloc@PLT
                leaq
                63(%rax), %r10
                andq
                $-64, %rbx
                andq
                $-64, %r15
                movq
                %rax, %r14
                andq
                $-64, %r10
                testl
                %ebp, %ebp
                je
                .L22
                movl
                (%rsp), %esi
                xorl
                %eax, %eax
                xorl
                %r11d, %r11d
                .p2align 4,,10
                .p2align 3
.L24:
                leal
                (%rsi,%rax),
%ecx
                .p2align 4,,10
                .p2align 3
.L23:
                movl
                %eax, %edx
                addl
                $1, %eax
                cmpl
                %ecx, %eax
                movl
                $0, (%rbx,
%rdx,4)
                movl
                $1,
(%r15,%rdx,4)
                movl
                $2,
(%r10,%rdx,4)
                jne
                .L23
                leal
                1(%r11), %edx
                cmpl
                %edx, %esi
                je
                .L34
                movl
                %edx, %r11d
                jmp
                .L24
                .p2align 4,,10
                .p2align 3
.L34:
                leaq
                32(%rsp), %r9
                leaq
                16(%rsp), %r8
                movq
                addl
                %edx, %ecx
                cmpq
                %r13, %rax
                movl
                %ecx, (%rsi)
                jne
                .L19
                addq
                $4, %rsi
                cmpq
                %rsi, %r10
                jne
                .L22
                addq
                $8, %r11
                cmpq
                %r14, %r11
                jne
                .L27
                leaq
                48(%rsp), %rsi
                xorl
                %edi, %edi
                call
clock_gettime@PLT
                cmpl
                $14, (%rsp)
                jbe
                .L18
                movq
                8(%rsp), %rdi
                pxor
                %xmm0, %xmm0
                pxor
                %xmm1, %xmm1
                movq
                (%rbx), %rdx
                leaq
                .LC6(%rip),
%rsi
                movq
                (%rbx,%rdi,8),
%rax
                movl
                (%rdx), %edx
                movl
                (%rax,%rdi,4),
%ecx
                movq
                56(%rsp), %rax
                movl
                $1, %edi
                subq
                40(%rsp), %rax
                cvtsi2sdq
                %rax, %xmm0
                movq
                48(%rsp), %rax
                subq
                32(%rsp), %rax
                cvtsi2sdq
                %rax, %xmm1
                movl
                $1, %eax
                divsd
                .LC5(%rip),
%xmm0
                addsd
                %xmm1, %xmm0
                call
__printf_chk@PLT
.L24:
                movl
                20(%rsp), %eax
                xorl
                %r13d, %r13d
                leaq
                8(,%rax,8),
%r14
                .p2align 4,,10
                .p2align 3
.L26:
                movq
                (%rbx,%r13),
%rdi
                call
                free@PLT
                movq
                0(%rbp,%r13),
%rdi
                call
                free@PLT
                movq
                (%r12,%r13),
%rdi
                addq
                $8, %r13
                call
                free@PLT
                cmpq
                %r14, %r13
                jne
                .L26
.L25:
                movq
                %rbx, %rdi
                call
                free@PLT
                movq
                %rbp, %rdi
```

```
                .cfi_restore_st
ate
                leaq
                32(%rsp), %rsi
                xorl
                %edi, %edi
                call
clock_gettime@PLT
                leaq
                48(%rsp), %rsi
                xorl
                %edi, %edi
                call
clock_gettime@PLT
.L18:
                leaq
                .LC2(%rip),
%rdi
                call
                puts@PLT
                movq
                8(%rsp), %r15
                movq
                %rbp, %rsi
                movl
                %r15d, %edi
                call
                printMatriz
                leaq
                .LC3(%rip),
%rdi
                call
                puts@PLT
                movq
                %r12, %rsi
                movl
                %r15d, %edi
                call
                printMatriz
                leaq
                .LC4(%rip),
%rdi
                call
                puts@PLT
                movq
                %rbx, %rsi
                movl
                %r15d, %edi
                call
                printMatriz
                jmp
                .L24
.L35:
                call
__stack_chk_fail@PLT
.L34:
                movq
                stderr(%rip),
%rcx
                leaq
                .LC1(%rip),
%rdi
                movl
                $40, %edx
                movl
                $1, %esi
                call
                fwrite@PLT
                movl
                $1, %edi
                call
                exit@PLT
                .cfi_endproc
.LFE53:
                .size
                main, .-main
                .section
                .rodata.cst8,"a
M",@progbits,8
                .align 8
.LC5:
                .long
                0
                .long
                1104006501
                .ident
                "GCC: (Ubuntu
7.3.0-16ubuntu3) 7.3.0"
                .section
                .note.GNU-
stack,"",@progbits
```

```
                %r10, %rdx
                movl
                %ebp, %ecx
                movq
                %r15, %rsi
                movq
                %rbx, %rdi
                movl
                %r11d, 12(%rsp)
                movq
                %r10, (%rsp)
                call
                pmm
                cmpl
                $14, %ebp
                movq
                (%rsp), %r10
                movl
                12(%rsp), %r11d
                jbe
                .L27
                leal
                (%r11,%r11),
%eax
                pxor
                %xmm0, %xmm0
                pxor
                %xmm1, %xmm1
                movl
                (%rbx), %edx
                movl
                (%rbx,%rax,4),
%ecx
                movq
                40(%rsp), %rax
                leaq
                .LC6(%rip),
%rsi
                subq
                24(%rsp), %rax
                movl
                $1, %edi
                cvtsi2sdq
                %rax, %xmm0
                movq
                32(%rsp), %rax
                subq
                16(%rsp), %rax
                cvtsi2sdq
                %rax, %xmm1
                movl
                $1, %eax
                divsd
                .LC5(%rip),
%xmm0
                addsd
                %xmm1, %xmm0
                call
__printf_chk@PLT
.L26:
                movq
                %r12, %rdi
                call
                free@PLT
                movq
                %r13, %rdi
                call
                free@PLT
                movq
                %r14, %rdi
                call
                free@PLT
                xorl
                %eax, %eax
                movq
                56(%rsp), %rdi
                xorq
                %fs:40, %rdi
                jne
                .L35
                addq
                $72, %rsp
                .cfi_remember_s
tate
                .cfi_def_cfa_of
fset 56
                popq
                %rbx
                .cfi_def_cfa_of
fset 48
                popq
                %rbp
                .cfi_def_cfa_of
fset 40
                popq
                %r12
                .cfi_def_cfa_of
fset 32
                popq
                %r13
                .cfi_def_cfa_of
fset 24
                popq
                %r14
                .cfi_def_cfa_of
fset 16
                popq
                %r15
                .cfi_def_cfa_of
fset 8
                ret
.L22:
```

```
                call
                free@PLT
                movq
                %r12, %rdi
                call
                free@PLT
                xorl
                %eax, %eax
                movq
                72(%rsp), %rbx
                xorq
                %fs:40, %rbx
                jne
                .L39
                addq
                $88, %rsp
                .cfi_remember_s
tate
                .cfi_def_cfa_of
fset 56
                popq
                %rbx
                .cfi_def_cfa_of
fset 48
                popq
                %rbp
                .cfi_def_cfa_of
fset 40
                popq
                %r12
                .cfi_def_cfa_of
fset 32
                popq
                %r13
                .cfi_def_cfa_of
fset 24
                popq
                %r14
                .cfi_def_cfa_of
fset 16
                popq
                %r15
                .cfi_def_cfa_of
fset 8
                ret
.L13:
                .cfi_restore_st
ate
                leaq
                32(%rsp), %rsi
                xorl
                %edi, %edi
                call
clock_gettime@PLT
                leaq
                48(%rsp), %rsi
                xorl
                %edi, %edi
                call
clock_gettime@PLT
.L18:
                leaq
                .LC2(%rip),
%rdi
                call
                puts@PLT
                movq
                (%rsp), %r15
                movq
                %rbp, %rsi
                movl
                %r15d, %edi
                call
                printMatriz
                leaq
                .LC3(%rip),
%rdi
                call
                puts@PLT
                movq
                %r12, %rsi
                movl
                %r15d, %edi
                call
                printMatriz
                leaq
                .LC4(%rip),
%rdi
                call
                puts@PLT
                movl
                %r15d, %edi
                movq
                %rbx, %rsi
                call
                printMatriz
                leal
                -1(%r15), %edi
                testl
                %r15d, %r15d
                movl
                %edi, 20(%rsp)
                jne
                .L24
                jmp
                .L25
.L39:
                call
__stack_chk_fail@PLT
.L38:
```

```
                    .cfi_restore_st
ate
                    leaq
32(%rsp), %r9
                    leaq
16(%rsp), %r8
                    movq
%r10, %rdx
                    xorl
%ecx, %ecx
                    movq
%r15, %rsi
                    movq
%rbx, %rdi
                    movq
%r10, (%rsp)
                    call
pmm
                    movq
(%rsp), %r10
.L27:
                    leaq
.LC2(%rip),
%rdi
                    movq
%r10, (%rsp)
                    call
puts@PLT
                    movq
%r15, %rsi
                    movl
%ebp, %edi
                    call
printMatriz
                    leaq
.LC3(%rip),
%rdi
                    call
puts@PLT
                    movq
(%rsp), %r10
                    movl
%ebp, %edi
                    movq
%r10, %rsi
                    call
printMatriz
                    leaq
.LC4(%rip),
%rdi
                    call
puts@PLT
                    movq
%rbx, %rsi
                    movl
%ebp, %edi
                    call
printMatriz
                    jmp
.L26
.L35:
                    call

__stack_chk_fail@PLT
.L33:
                    movq
stderr(%rip),
%rcx
                    leaq
.LC1(%rip),
%rdi
                    movl
$40, %edx
                    movl
$1, %esi
                    call
fwrite@PLT
                    movl
$1, %edi
                    call
exit@PLT
                    .cfi_endproc
.LFE54:
                    .size
main, .-main
                    .section
.rodata.cst8,"a
M",@progbits,8
                    .align 8
.LC5:
                    .long
0
                    .long
1104006501
                    .ident
"GCC: (Ubuntu
7.3.0-16ubuntu3) 7.3.0"
                    .section
.note.GNU-
stack,"",@progbits
```

```
                    movq
stderr(%rip),
%rcx
                    leaq
.LC1(%rip),
%rdi
                    movl
$40, %edx
                    movl
$1, %esi
                    call
fwrite@PLT
                    movl
$1, %edi
                    call
exit@PLT
                    .cfi_endproc
.LFE53:
                    .size
main, .-main
                    .section
.rodata.cst8,"a
M",@progbits,8
                    .align 8
.LC5:
                    .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>

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

int main() {
  int ii, i, X1, X2;
  int R[40000];

  struct timespec cgt1, cgt2;
  double ncgt;

  clock_gettime(CLOCK_REALTIME, &cgt1);
  for (ii = 1; 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;
    }
  }

  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[39999] = %i\n", R[0], R[39999]);
  printf("\nTiempo (seg.) = %11.9f\n", ncgt);

  return 0;
}
```

1.1. MODIFICACIONES REALIZADAS (al menos dos modificaciones):

Modificación a) –explicación-: Se unen los bucles internos para realizar la mitad de iteraciones a pesar de hacer más calculos.

Modificación b) –explicación-: Eliminar uno de los bucles for, utilizando uno solo.

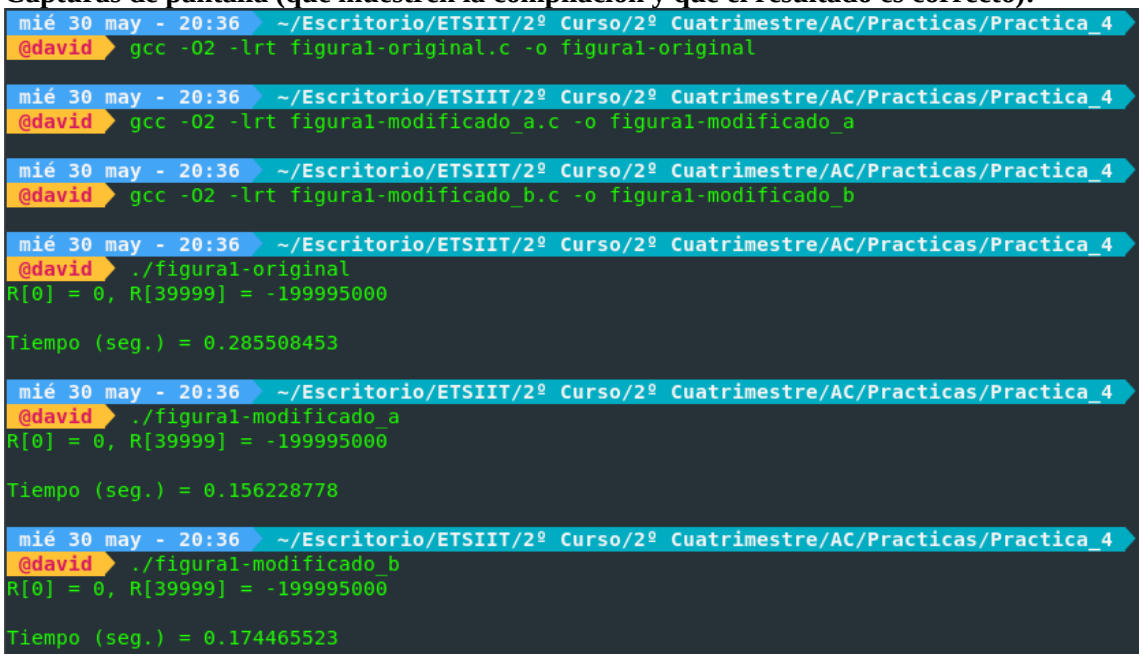
1.1. CÓDIGOS FUENTE MODIFICACIONES

a) Captura figura1**`-modificado_a.c`**

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

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

void Funcion(int *R) {
  int ii, i, X1, X2;

  for (ii = 1; ii <= 40000; ii++) {
    X1 = 0;
    X2 = 0;

    for (i = 0; i < 5000; 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;
    }
  }
}

int main() {
  int R[40000];

  struct timespec cgt1, cgt2;
  double ncgt;

  clock_gettime(CLOCK_REALTIME, &cgt1);

  Funcion(R);

  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[39999] = %i\n", R[0], R[39999]);
  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>

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

void Funcion(int *R) {
  int ii, i, X1, X2;

  for (ii = 1; ii <= 40000; ii++) {
    X1 = 0;
    X2 = 0;

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

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

int main() {
  int R[40000];

  struct timespec cgt1, cgt2;
  double ncgt;

  clock_gettime(CLOCK_REALTIME, &cgt1);

  Funcion(R);

  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[39999] = %i\n", R[0], R[39999]);
  printf("\nTiempo (seg.) = %11.9f\n", ncgt);

  return 0;
}
```

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

```
mié 30 may - 20:36  ~/Escritorio/ETSIIT/2º Curso/2º Cuatrimestre/AC/Practicas/Practica_4
@david  gcc -O2 -lrt figura1-original.c -o figura1-original

mié 30 may - 20:36  ~/Escritorio/ETSIIT/2º Curso/2º Cuatrimestre/AC/Practicas/Practica_4
@david  gcc -O2 -lrt figura1-modificado_a.c -o figura1-modificado_a

mié 30 may - 20:36  ~/Escritorio/ETSIIT/2º Curso/2º Cuatrimestre/AC/Practicas/Practica_4
@david  gcc -O2 -lrt figura1-modificado_b.c -o figura1-modificado_b

mié 30 may - 20:36  ~/Escritorio/ETSIIT/2º Curso/2º Cuatrimestre/AC/Practicas/Practica_4
@david  ./figura1-original
R[0] = 0, R[39999] = -199995000

Tiempo (seg.) = 0.285508453

mié 30 may - 20:36  ~/Escritorio/ETSIIT/2º Curso/2º Cuatrimestre/AC/Practicas/Practica_4
@david  ./figura1-modificado_a
R[0] = 0, R[39999] = -199995000

Tiempo (seg.) = 0.156228778

mié 30 may - 20:36  ~/Escritorio/ETSIIT/2º Curso/2º Cuatrimestre/AC/Practicas/Practica_4
@david  ./figura1-modificado_b
R[0] = 0, R[39999] = -199995000

Tiempo (seg.) = 0.174465523
```

1.1. TIEMPOS:

Modificación	-O2
Sin modificar	0.285508453
Modificación a)	0.156228778
Modificación b)	0.174465523

1.1. COMENTARIOS SOBRE LOS RESULTADOS:

Se puede observar que la modificación a es de la que mejor resultado se obtiene debido a que al unificar todos los cálculos en un solo bucle se reduce el número de iteraciones que se deben de realizar.

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   "figura1-original.c"
                .text
                .section        .rodata.str1.1,"aMS",@progbits,1
.LC1:
                .string "R[0] = %i, R[39999] = %i\n"
.LC2:
                .string "\nTiempo (seg.) = %11.9f\n"
                .section        .text.startup,"ax",@progbits
                .p2align 4,,15
                .globl  main
                .type   main, @function
main:
.LFB23:
                .cfi_startproc
                subq    $160072, %rsp
                .cfi_def_cfa_offset 160080
                xorl    %edi, %edi
                leaq    16(%rsp), %rsi
                movq    %fs:40, %rax
                movq    %rax, 160056(%rsp)
                xorl    %eax, %eax
                call    clock_gettime@PLT
                leaq    40004+s(%rip), %r9
                leaq    48(%rsp), %r11
                movl    $1, %r10d
                leaq    -4(%r9), %r8
                .p2align 4,,10
                .p2align 3
.L2:
                leaq    s(%rip), %rax
                movl    %r10d, %edi
                xorl    %esi, %esi
                .p2align 4,,10
                .p2align 3
.L3:
                movl    (%rax), %edx
                addq    $8, %rax
                leal    (%rdi,%rdx,2), %edx
                addl    %edx, %esi
                cmpq    %r8, %rax
                jne     .L3
                leaq
```

pmm-secuencial-modificado_b.s

```
                .file   "figura1-modificado_a.c"
                .text
                .p2align 4,,15
                .globl  Funcion
                .type   Funcion, @function
Funcion:
.LFB23:
                .cfi_startproc
                leaq    40000+s(%rip), %r8
                movl    $1, %r9d
                .p2align 4,,10
                .p2align 3
.L2:
                leaq    s(%rip), %rax
                movl    %r9d, %edx
                xorl    %ecx, %ecx
                xorl    %r10d, %r10d
                .p2align 4,,10
                .p2align 3
.L3:
                movl    (%rax), %esi
                addq    $32, %rax
                leal    (%rdx,%rsi,2), %r11d
                movl    -28(%rax), %esi
                addl    %r10d, %r11d
                leal    (%rsi,%rsi,2), %esi
                subl    %edx, %esi
                addl    %esi, %ecx
                movl    -24(%rax), %esi
                leal    (%rdx,%rsi,2), %r10d
                movl    -20(%rax), %esi
                addl    %r10d, %r11d
                leal    (%rsi,%rsi,2), %esi
                subl    %edx, %esi
                addl    %ecx, %esi
                movl    -16(%rax), %ecx
                leal    (%rdx,%rcx,2), %r10d
                movl    -12(%rax), %ecx
                addl    %r11d, %r10d
                leal    (%rcx,%rcx,2), %ecx
                subl
```

pmm-secuencial-modificado_c.s

```
                .file   "figura1-modificado_b.c"
                .text
                .p2align 4,,15
                .globl  Funcion
                .type   Funcion, @function
Funcion:
.LFB23:
                .cfi_startproc
                leaq    40000+s(%rip), %r9
                movl    $1, %r10d
                .p2align 4,,10
                .p2align 3
.L2:
                leaq    s(%rip), %rax
                movl    %r10d, %esi
                xorl    %edx, %edx
                xorl    %ecx, %ecx
                .p2align 4,,10
                .p2align 3
.L3:
                movl    (%rax), %r8d
                addq    $8, %rax
                leal    (%rsi,%r8,2), %r8d
                addl    %r8d, %ecx
                movl    -4(%rax), %r8d
                leal    (%rdx,%r8,2), %edx
                subl    %esi, %edx
                cmpq    %rax, %r9
                jne     .L3
                cmpl    %edx, %ecx
                cmovl   %ecx, %edx
                movl    %edx, (%rdi, %r10,4)
                addq    $1, %r10
                cmpq    $40001, %r10
                jne     .L2
                rep ret
                .cfi_endproc
.LFE23:
                .size   Funcion, .-Funcion
                .section        .rodata.str1.1,"aMS",@progbits,1
.LC1:
                .string "R[0] = %i, R[39999] = %i\n"
.LC2:
```

```
                4+s(%rip), %rax
                xorl
                %ecx, %ecx
                .p2align 4,,10
                .p2align 3
.L4:
                movl
                (%rax), %edx
                addq
                $8, %rax
                leal
                (%rdx,%rdx,2),
%edx
                subl
                %edi, %edx
                addl
                %edx, %ecx
                cmpq
                %rax, %r9
                jne
                .L4
                cmpl
                %esi, %ecx
                cmovg
                %esi, %ecx
                movl
                %ecx,
(%r11,%r10,4)
                addq
                $1, %r10
                cmpq
                $40001, %r10
                jne
                .L2
                leaq
                32(%rsp), %rsi
                xorl
                %edi, %edi
                call
clock_gettime@PLT
                movq
                40(%rsp), %rax
                subq
                24(%rsp), %rax
                leaq
                .LC1(%rip),
%rsi
                pxor
                %xmm0, %xmm0
                movl
                160044(%rsp),
%ecx
                pxor
                %xmm1, %xmm1
                movl
                48(%rsp), %edx
                movl
                $1, %edi
                cvtsi2sdq
                %rax, %xmm0
                movq
                32(%rsp), %rax
                subq
                16(%rsp), %rax
                cvtsi2sdq
                %rax, %xmm1
                xorl
                %eax, %eax
                divsd
                .LC0(%rip),
%xmm0
                addsd
                %xmm1, %xmm0
                movsd
                %xmm0, 8(%rsp)
                call
__printf_chk@PLT
                leaq
                .LC2(%rip),
%rsi
                movsd
                8(%rsp), %xmm0
                movl
                $1, %edi
                movl
                $1, %eax
                call
__printf_chk@PLT
                xorl
                %eax, %eax
                movq
                160056(%rsp),
%rsi
                xorq
                %fs:40, %rsi
                jne
                .L13
                addq
                $160072, %rsp
                .cfi_remember_s
tate
                .cfi_def_cfa_of
fset 8
                ret
.L13:
                .cfi_restore_st
ate
                call
__stack_chk_fail@PLT
                %edx, %ecx
                addl
                %ecx, %esi
                movl
                -8(%rax), %ecx
                leal
                (%rdx,%rcx,2),
%ecx
                addl
                %ecx, %r10d
                movl
                -4(%rax), %ecx
                leal
                (%rcx,%rcx,2),
%ecx
                subl
                %edx, %ecx
                addl
                %esi, %ecx
                cmpq
                %rax, %r8
                jne
                .L3
                cmpl
                %ecx, %r10d
                cmovl
                %r10d, %ecx
                movl
                %ecx, (%rdi,
%r9,4)
                addq
                $1, %r9
                cmpq
                $40001, %r9
                jne
                .L2
                rep ret
                .cfi_endproc
.LFE23:
                .size
                Funcion, .-
Funcion
                .section
                .rodata.str1.1,
"aMS",@progbits,1
.LC1:
                .string
                "R[0] = %i,
R[39999] = %i\n"
.LC2:
                .string
                "\nTiempo
(seg.) = %11.9f\n"
                .section
                .text.startup,"
ax",@progbits
                .p2align 4,,15
                .globl
                main
                .type
                main, @function
main:
.LFB24:
                .cfi_startproc
                subq
                $160072, %rsp
                .cfi_def_cfa_of
fset 160080
                xorl
                %edi, %edi
                leaq
                16(%rsp), %rsi
                movq
                %fs:40, %rax
                movq
                %rax,
160056(%rsp)
                xorl
                %eax, %eax
                call
clock_gettime@PLT
                leaq
                48(%rsp), %rdi
                call
                Funcion
                leaq
                32(%rsp), %rsi
                xorl
                %edi, %edi
                call
clock_gettime@PLT
                movq
                40(%rsp), %rax
                subq
                24(%rsp), %rax
                leaq
                .LC1(%rip),
%rsi
                pxor
                %xmm0, %xmm0
                movl
                48(%rsp), %edx
                pxor
                %xmm1, %xmm1
                movl
                160044(%rsp),
%ecx
                movl
                $1, %edi
                cvtsi2sdq
                %rax, %xmm0
                movq
                32(%rsp), %rax
                subq
                16(%rsp), %rax
                cvtsi2sdq
                %rax, %xmm1
                xorl
                %eax, %eax
                divsd
                .LC0(%rip),
%xmm0
                addsd
                %xmm1, %xmm0
                movsd
                %xmm0, 8(%rsp)
                call
__printf_chk@PLT
                movsd
                8(%rsp), %xmm0
                leaq
                .LC2(%rip),
%rsi
                movl
                $1, %edi
                movl
                $1, %eax
                call
__printf_chk@PLT
                movq
                160056(%rsp),
%rdx
                xorq
                %fs:40, %rdx
                jne
                .L12
                xorl
                %eax, %eax
                addq
                $160072, %rsp
                .cfi_remember_s
tate
                .cfi_def_cfa_of
fset 8
                ret
.L12:
                .cfi_restore_st
ate
                call
```

```
                .cfi_endproc
.LFE23:
                .size
                main, .-main
                .comm
                s,40000,32
                .section
                .rodata.cst8,"a
M",@progbits,8
                .align 8
.LC0:
                .long
                0
                .long
                1104006501
                .ident
                "GCC: (Ubuntu
7.3.0-16ubuntu3) 7.3.0"
                .section
                .note.GNU-
stack,"",@progbits
```

```
                movq
                32(%rsp), %rax
                subq
                16(%rsp), %rax
                cvtsi2sdq
                %rax, %xmm1
                xorl
                %eax, %eax
                divsd
                .LC0(%rip),
%xmm0
                addsd
                %xmm1, %xmm0
                movsd
                %xmm0, 8(%rsp)
                call

__printf_chk@PLT
                movsd
                8(%rsp), %xmm0
                leaq
                .LC2(%rip),
%rsi
                movl
                $1, %edi
                movl
                $1, %eax
                call

__printf_chk@PLT
                movq
                160056(%rsp),
%rdx
                xorq
                %fs:40, %rdx
                jne
                .L12
                xorl
                %eax, %eax
                addq
                $160072, %rsp
                .cfi_remember_s
tate
                .cfi_def_cfa_of
fset 8
                ret
.L12:
                .cfi_restore_st
ate
                call

__stack_chk_fail@PLT
                .cfi_endproc
.LFE24:
                .size
                main, .-main
                .comm
                s,40000,32
                .section
                .rodata.cst8,"a
M",@progbits,8
                .align 8
.LC0:
                .long
                0
                .long
                1104006501
                .ident
                "GCC: (Ubuntu
7.3.0-16ubuntu3) 7.3.0"
                .section
                .note.GNU-
stack,"",@progbits
```

```
__stack_chk_fail@PLT
                .cfi_endproc
.LFE24:
                .size
                main, .-main
                .comm
                s,40000,32
                .section
                .rodata.cst8,"a
M",@progbits,8
                .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>

void DAXPY(int *y, int *x, int a, unsigned n, struct timespec *cgt1,
           struct timespec *cgt2) {
  clock_gettime(CLOCK_REALTIME, cgt1);
  unsigned i;
  for (i = 0; i < n; i++) {
    y[i] += a * x[i];
  }
  clock_gettime(CLOCK_REALTIME, cgt2);
}

int main(int argc, char *argv[]) {
  if (argc < 3) {
    fprintf(stderr, "ERROR: falta tam del vector y constante\n");
    exit(1);
  }

  int *y, *x;
  int a = strtol(argv[2], NULL, 10);
  unsigned n = strtol(argv[1], NULL, 10);
  y = (int *)malloc(n * sizeof(int));
  x = (int *)malloc(n * sizeof(int));

  unsigned i;
  for (i = 0; i < n; i++) {
    y[i] = i + 2;
    x[i] = i * 2;
  }

  struct timespec cgt1, cgt2;
  double ncgt;

  DAXPY(y, x, a, n, &cgt1, &cgt2);

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

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

  free(y);
  free(x);

  return 0;
}
```

Tiempos ejec.	-O0	-Os	-O2	-O3
	0.355463460	0.125826016	0.103762091	0.082723644

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

```
mié 30 may - 21:02  ~/Escritorio/ETSIIT/2º Curso/2º Cuatrimestre/AC/Practicas/Practica_4
@david  gcc -O0 -lrt daxpy.c -o daxpyO0

mié 30 may - 21:02  ~/Escritorio/ETSIIT/2º Curso/2º Cuatrimestre/AC/Practicas/Practica_4
@david  gcc -Os -lrt daxpy.c -o daxpyOs

mié 30 may - 21:02  ~/Escritorio/ETSIIT/2º Curso/2º Cuatrimestre/AC/Practicas/Practica_4
@david  gcc -O2 -lrt daxpy.c -o daxpyO2

mié 30 may - 21:02  ~/Escritorio/ETSIIT/2º Curso/2º Cuatrimestre/AC/Practicas/Practica_4
@david  gcc -O3 -lrt daxpy.c -o daxpyO3

mié 30 may - 21:02  ~/Escritorio/ETSIIT/2º Curso/2º Cuatrimestre/AC/Practicas/Practica_4
@david  ./daxpyO0 100000000 400
y[0] = 2, y[99999999] = -1504379423

Tiempo (seg.) = 0.355463460

mié 30 may - 21:03  ~/Escritorio/ETSIIT/2º Curso/2º Cuatrimestre/AC/Practicas/Practica_4
@david  ./daxpyOs 100000000 400
y[0] = 2, y[99999999] = -1504379423

Tiempo (seg.) = 0.125826016

mié 30 may - 21:03  ~/Escritorio/ETSIIT/2º Curso/2º Cuatrimestre/AC/Practicas/Practica_4
@david  ./daxpyO2 100000000 400
y[0] = 2, y[99999999] = -1504379423

Tiempo (seg.) = 0.103762091

mié 30 may - 21:03  ~/Escritorio/ETSIIT/2º Curso/2º Cuatrimestre/AC/Practicas/Practica_4
@david  ./daxpyO3 100000000 400
y[0] = 2, y[99999999] = -1504379423

Tiempo (seg.) = 0.082723644
```

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)

daxpyO0.s	daxpyOs.s	daxpyO2.s	daxpyO3.s
.file "daxpy.c" .text .globl DAXPY .type DAXPY, @function DAXPY: .LFB5: .cfi_startproc pushq %rbp .cfi_def_cfa_offset 16 .cfi_offset 6, -16 movq %rsp, %rbp .cfi_def_	.file "daxpy.c" .text .globl DAXPY .type DAXPY, @function DAXPY: .LFB23: .cfi_startproc pushq %r13 .cfi_def_cfa_offset 16 .cfi_offset 13, -16 pushq %r12 .cfi_def	.file "daxpy.c" .text .p2align 4,,15 .globl DAXPY .type DAXPY, @function DAXPY: .LFB41: .cfi_startproc pushq %r14 .cfi_def_cfa_offset 16 .cfi_offset 14, -16 pushq %r13	.file "daxpy.c" .text .p2align 4,,15 .globl DAXPY .type DAXPY, @function DAXPY: .LFB41: .cfi_startproc pushq %r14 .cfi_def_cfa_offset 16 .cfi_offset 14, -16 pushq %r13

```
cfa_register 6
        subq    $64, %rsp
        movq    %rdi, -24(%rbp)
        movq    %rsi, -32(%rbp)
        movl    %edx, -36(%rbp)
        movl    %ecx, -40(%rbp)
        movq    %r8, -48(%rbp)
        movq    %r9, -56(%rbp)
        movq    -48(%rbp), %rax
        movq    %rax, %rsi
        movl    $0, %edi
        call    clock_gettime@PLT
        movl    $0, -4(%rbp)
        jmp     .L2
.L3:
        movl    -4(%rbp), %eax
        leaq    0(,%rax,4), %rdx
        movq    -24(%rbp), %rax
        addq    %rdx, %rax
        movl    (%rax), %ecx
        movl    -4(%rbp), %eax
        leaq    0(,%rax,4), %rdx
        movq    -32(%rbp), %rax
        addq    %rdx, %rax
        movl    (%rax), %eax
        imull   -36(%rbp), %eax
        movl    %eax, %edx
        movl    -4(%rbp), %eax
        leaq    0(,%rax,4), %rsi
        movq    -24(%rbp), %rax
        addq    %rsi, %rax
        addl    %ecx,
```

```
_cfa_offset 24
        .cfi_offset 12, -24
        movq    %rsi, %r12
        pushq   %rbp
        .cfi_def_cfa_offset 32
        .cfi_offset 6, -32
        pushq   %rbx
        .cfi_def_cfa_offset 40
        .cfi_offset 3, -40
        movq    %r8, %rsi
        movq    %rdi, %rbx
        xorl    %edi, %edi
        movl    %edx, %r13d
        subq    $24, %rsp
        .cfi_def_cfa_offset 64
        movl    %ecx, %ebp
        movq    %r9, 8(%rsp)
        call    clock_gettime@PLT
        movq    8(%rsp), %r9
        xorl    %eax, %eax
.L2:
        cmpl    %eax, %ebp
        jbe     .L6
        movl    (%r12,%rax,4), %edx
        imull   %r13d, %edx
        addl    %edx, (%rbx,%rax,4)
        incq    %rax
        jmp     .L2
.L6:
        addq    $24, %rsp
        .cfi_def_cfa_offset 40
        movq    %r9, %rsi
        xorl    %edi, %edi
        popq    %rbx
        .cfi_def_cfa_offset 32
        popq    %rbp
        .cfi_def
```

```
        .cfi_def_cfa_offset 24
        .cfi_offset 13, -24
        movl    %ecx, %r14d
        pushq   %r12
        .cfi_def_cfa_offset 32
        .cfi_offset 12, -32
        pushq   %rbp
        .cfi_def_cfa_offset 40
        .cfi_offset 6, -40
        movq    %rsi, %rbp
        pushq   %rbx
        .cfi_def_cfa_offset 48
        .cfi_offset 3, -48
        movq    %r8, %rsi
        movq    %rdi, %rbx
        xorl    %edi, %edi
        movl    %edx, %r12d
        movq    %r9, %r13
        call    clock_gettime@PLT
        testl   %r14d, %r14d
        je      .L2
        leal    -1(%r14), %eax
        leaq    4(,%rax,4), %rdx
        xorl    %eax, %eax
        .p2align 4,,10
        .p2align 3
.L3:
        movl    0(%rbp, %rax), %ecx
        imull   %r12d, %ecx
        addl    %ecx, (%rbx,%rax)
        addq    $4, %rax
        cmpq    %rax, %rdx
        jne     .L3
.L2:
        popq    %rbx
        .cfi_def_cfa_offset 40
        movq    %r13, %rsi
        xorl    %edi, %edi
        popq    %rbp
        .cfi_def_cfa_offset 32
        popq    %r12
```

```
        .cfi_def_cfa_offset 24
        .cfi_offset 13, -24
        movl    %ecx, %r14d
        pushq   %r12
        .cfi_def_cfa_offset 32
        .cfi_offset 12, -32
        pushq   %rbp
        .cfi_def_cfa_offset 40
        .cfi_offset 6, -40
        movq    %rsi, %r12
        pushq   %rbx
        .cfi_def_cfa_offset 48
        .cfi_offset 3, -48
        movq    %r8, %rsi
        movq    %rdi, %rbx
        xorl    %edi, %edi
        movl    %edx, %r13d
        movq    %r9, %rbp
        subq    $16, %rsp
        .cfi_def_cfa_offset 64
        call    clock_gettime@PLT
        testl   %r14d, %r14d
        je      .L2
        leaq    16(%r12), %rax
        cmpq    %rax, %rbx
        leaq    16(%rbx), %rax
        setnb   %dl
        cmpq    %rax, %r12
        setnb   %al
        orb     %al, %dl
        je      .L3
        cmpl    $6, %r14d
        jbe     .L3
        movq    %rbx, %rsi
        xorl    %edi, %edi
        shrq    $2, %rsi
        negq    %rsi
        andl
```

```
%edx
        movl    %edx, (%rax)
        addl    $1, -4(%rbp)
.L2:
        movl    -4(%rbp), %eax
        cmpl    -40(%rbp), %eax
        jb      .L3
        movq    -56(%rbp), %rax
        movq    %rax, %rsi
        movl    $0, %edi
        call    clock_gettime@PLT
        nop
        leave
        .cfi_def_cfa 7, 8
        ret
        .cfi_endproc
.LFE5:
        .size   DAXPY, .-DAXPY
        .section        .rodata
        .align 8
.LC0:
        .string "ERROR: falta tam del vector y constante\n"
.LC2:
        .string "y[0] = %i, y[%i] = %i\n"
.LC3:
        .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
        addq    $-128, %rsp
        movl    %edi, -100(%rbp)
        movq    %rsi, -112(%rbp)
        movq    %fs:40, %rax
        movq    %rax, -8(%rbp)
        xorl    %eax, %eax
        movl    $0, %edi
        call    clock_gettime@PLT
        movl    $0, -4(%rbp)
        jmp     .L2
.L3:
        movl    -4(%rbp), %eax
        cltq
        movq    $0x0, -80(%rbp,%rax,8)
        addl    $1, -4(%rbp)
.L2:
        movl    -4(%rbp), %eax
        cmpl    -40(%rbp), %eax
        jb      .L3
        movq    -56(%rbp), %rax
        movq    %rax, %rsi
        movl    $0, %edi
        call    clock_gettime@PLT
        movl    $0, -4(%rbp)
        jmp     .L4
.L5:
        movl    -4(%rbp), %eax
        cltq
        movq    -80(%rbp,%rax,8), %rax
        movq    %rax, %rdx
        movl    -4(%rbp), %eax
        movl    %eax, %esi
        movl    $.LC2, %edi
        movl    $0, %eax
        call    printf
        addl    $1, -4(%rbp)
.L4:
        movl    -4(%rbp), %eax
        cmpl    -40(%rbp), %eax
        jb      .L5
        movq    -56(%rbp), %rax
        movq    %rax, %rsi
        movl    $0, %edi
        call    clock_gettime@PLT
        movl    $0, %eax
        movq    -8(%rbp), %rdx
        xorq    %fs:40, %rdx
        je      .L6
        call    __stack_chk_fail
.L6:
        leave
        .cfi_def_cfa 7, 8
        ret
        .cfi_endproc
.LFE6:
        .size   main, .-main
        .ident  "GCC: (Ubuntu 7.5.0-3ubuntu1~18.04) 7.5.0"
        .section        .note.GNU-stack,"",@progbits
```

```
_cfa_offset 24
        popq    %r12
        .cfi_def_cfa_offset 16
        popq    %r13
        .cfi_def_cfa_offset 8
        jmp     clock_gettime@PLT
        .cfi_endproc
.LFE23:
        .size   DAXPY, .-DAXPY
        .section        .rodata.str1.1,"aMS",@progbits,1
.LC0:
        .string "ERROR: falta tam del vector y constante\n"
.LC2:
        .string "y[0] = %i, y[%i] = %i\n"
.LC3:
        .string "\nTiempo (seg.) = %11.9f\n"
        .section        .text.startup,"ax",@progbits
        .globl  main
        .type   main, @function
main:
.LFB24:
        .cfi_startproc
        pushq   %r14
        .cfi_def_cfa_offset 16
        .cfi_offset 14, -16
        pushq   %r13
        .cfi_def_cfa_offset 24
        .cfi_offset 13, -24
        pushq   %r12
        .cfi_def_cfa_offset 32
        .cfi_offset 12, -32
        pushq   %rbp
        .cfi_def_cfa_offset 40
        .cfi_offset 6, -40
        pushq   %rbx
        .cfi_def_cfa_offset 48
        .cfi_offset 3, -48
        subq    $64, %rsp
        .cfi_def_cfa_offset 112
        movq    %fs:40, %rax
        movq    %rax, 56(%rsp)
        xorl    %eax, %eax
        movl    $0, %edi
        call    clock_gettime@PLT
        movl    $0, %eax
        movq    -8(%rbp), %rdx
        xorq    %fs:40, %rdx
        je      .L6
        call    __stack_chk_fail
.L6:
        leave
        .cfi_def_cfa 7, 8
        ret
        .cfi_endproc
.LFE6:
        .size   main, .-main
        .ident  "GCC: (Ubuntu 7.5.0-3ubuntu1~18.04) 7.5.0"
        .section        .note.GNU-stack,"",@progbits
```

```
        .cfi_def_cfa_offset 24
        popq    %r13
        .cfi_def_cfa_offset 16
        popq    %r14
        .cfi_def_cfa_offset 8
        jmp     clock_gettime@PLT
        .cfi_endproc
.LFE41:
        .size   DAXPY, .-DAXPY
        .section        .rodata.str1.8,"aMS",@progbits,1
        .align 8
.LC0:
        .string "ERROR: falta tam del vector y constante\n"
        .section        .rodata.str1.1,"aMS",@progbits,1
.LC2:
        .string "y[0] = %i, y[%i] = %i\n"
.LC3:
        .string "\nTiempo (seg.) = %11.9f\n"
        .section        .text.startup,"ax",@progbits
        .p2align 4,,15
        .globl  main
        .type   main, @function
main:
.LFB42:
        .cfi_startproc
        pushq   %r14
        .cfi_def_cfa_offset 16
        .cfi_offset 14, -16
        pushq   %r13
        .cfi_def_cfa_offset 24
        .cfi_offset 13, -24
        pushq   %r12
        .cfi_def_cfa_offset 32
        .cfi_offset 12, -32
        pushq   %rbp
        .cfi_def_cfa_offset 40
        .cfi_offset 6, -40
        pushq   %rbx
        .cfi_def_cfa_offset 48
        .cfi_offset 3, -48
        subq    $64, %rsp
        .cfi_def_cfa_offset 112
        movq    %fs:40, %rax
        movq    %rax, 56(%rsp)
        xorl    %eax, %eax
        movl    $0, %edi
        call    clock_gettime@PLT
        movl    $0, %eax
        movq    -8(%rbp), %rdx
        xorq    %fs:40, %rdx
        je      .L6
        call    __stack_chk_fail
.L6:
        leave
        .cfi_def_cfa 7, 8
        ret
        .cfi_endproc
.LFE6:
        .size   main, .-main
        .ident  "GCC: (Ubuntu 7.5.0-3ubuntu1~18.04) 7.5.0"
        .section        .note.GNU-stack,"",@progbits
```

```
        $3, %esi
        je      .L4
        movl    (%r12), %eax
        movl    $1, %edi
        imull   %r13d, %eax
        addl    %eax, (%rbx)
        cmpl    $1, %esi
        je      .L4
        movl    4(%r12), %eax
        movl    $2, %edi
        imull   %r13d, %eax
        addl    %eax, 4(%rbx)
        cmpl    $3, %esi
        jne     .L4
        movl    8(%r12), %eax
        movl    $3, %edi
        imull   %r13d, %eax
        addl    %eax, 8(%rbx)
.L4:
        movl    %r13d, 12(%rsp)
        movl    %r14d, %r8d
        xorl    %eax, %eax
        movd    12(%rsp), %xmm4
        subl    %esi, %r8d
        movl    %esi, %esi
        salq    $2, %rsi
        movl    %r8d, %r9d
        xorl    %edx, %edx
        pshufd  $0, %xmm4, %xmm2
        leaq    (%rbx, %rsi), %rcx
        shrl    $2, %r9d
        addq    %r12, %rsi
        movdqa  %xmm2, %xmm3
        psrlq   $32, %xmm3
```

```
        %rax, -8(%rbp)
        xorl    %eax, %eax
        cmpl    $2, -100(%rbp)
        jg      .L5
        movq    stderr(%rip), %rax
        movq    %rax, %rcx
        movl    $40, %edx
        movl    $1, %esi
        leaq    .LC0(%rip), %rdi
        call    fwrite@PLT
        movl    $1, %edi
        call    exit@PLT
.L5:
        movq    -112(%rbp), %rax
        addq    $16, %rax
        movq    (%rax), %rax
        movl    $10, %edx
        movl    $0, %esi
        movq    %rax, %rdi
        call    strtol@PLT
        movl    %eax, -80(%rbp)
        movq    -112(%rbp), %rax
        addq    $8, %rax
        movq    (%rax), %rax
        movl    $10, %edx
        movl    $0, %esi
        movq    %rax, %rdi
        call    strtol@PLT
        movl    %eax, -76(%rbp)
        movl    -76(%rbp), %eax
        salq    $2, %rax
        movq    %rax, %rdi
        call    malloc@PLT
        movq    %rax, -72(%rbp)
        movl    56(%rsp)
        xorl    %eax, %eax
        cmpl    $2, %edi
        jg      .L8
        movq    stderr(%rip), %rsi
        leaq    .LC0(%rip), %rdi
        call    fputs@PLT
        movl    $1, %edi
        call    exit@PLT
.L8:
        movq    16(%rsi), %rdi
        movq    %rsi, %rbx
        movl    $10, %edx
        xorl    %esi, %esi
        call    strtol@PLT
        movq    8(%rbx), %rdi
        movl    $10, %edx
        xorl    %esi, %esi
        movq    %rax, %r13
        call    strtol@PLT
        movl    %eax, %r12d
        movq    %rax, %rbx
        movl    %eax, %r14d
        salq    $2, %r12
        movq    %r12, %rdi
        call    malloc@PLT
        movq    %r12, %rdi
        movq    %rax, %rbp
        call    malloc@PLT
        movq    %rax, %r12
        xorl    %eax, %eax
.L9:
        cmpl    %eax, %r14d
        movq    %fs:40, %rax
        movq    %rax, 56(%rsp)
        xorl    %eax, %eax
        cmpl    $2, %edi
        jle     .L18
        movq    16(%rsi), %rdi
        movq    %rsi, %rbx
        movl    $10, %edx
        xorl    %esi, %esi
        movl    $-1, %r13d
        call    strtol@PLT
        movq    8(%rbx), %rdi
        xorl    %esi, %esi
        movl    $10, %edx
        movq    %rax, %r12
        call    strtol@PLT
        movl    %eax, %ebp
        movq    %rax, %r14
        salq    $2, %rbp
        movq    %rbp, %rdi
        call    malloc@PLT
        movq    %rbp, %rdi
        movq    %rax, %rbx
        call    malloc@PLT
        testl   %r14d, %r14d
        movq    %rax, %rbp
        je      .L12
        leal    (%r14,%r13), %esi
        xorl    %eax, %eax
        movq    %rsi, %r13
        addq    $1, %rsi
        .p2align 4,,10
        .p2align 3
.L13:
        leal    2(%rax), %edx
        movl    %edx, (%rbx,%rax,4)
        leal    (%rax, %rax), %edx
        movl    %edx, 0(%rbp,%rax,4)
        addq    $1, %rax
        cmpq    %rsi, %rax
        .p2align 4,,10
        .p2align 3
.L6:
        movdqu  (%rsi, %rax), %xmm0
        addl    $1, %edx
        movdqa  %xmm0, %xmm1
        psrlq   $32, %xmm0
        pmuludq %xmm3, %xmm0
        pshufd  $8, %xmm0, %xmm0
        pmuludq %xmm2, %xmm1
        pshufd  $8, %xmm1, %xmm1
        punpckldq %xmm0, %xmm1
        movdqa  (%rcx, %rax), %xmm0
        paddd   %xmm1, %xmm0
        movaps  %xmm0, (%rcx,%rax)
        addq    $16, %rax
        cmpl    %edx, %r9d
        ja      .L6
        movl    %r8d, %edx
        andl    $-4, %edx
        cmpl    %edx, %r8d
        leal    (%rdi, %rdx), %eax
        je      .L2
        movl    %eax, %edx
        movl    (%r12,%rdx,4), %ecx
        imull   %r13d, %ecx
        addl    %ecx, (%rbx,%rdx,4)
        leal    1(%rax), %edx
        cmpl    %edx, %r14d
        jbe     .L2
        movl    (%r12,%rdx,4), %ecx
        imull   %r13d, %ecx
        addl    %ecx,
```

```
            -
76(%rbp), %eax
            salq
            $2, %rax
            movq
            %rax,
%rdi
            call
malloc@PLT
            movq
            %rax, -
64(%rbp)
            movl
            $0, -
84(%rbp)
            jmp
            .L6
.L7:
            movl
            -
84(%rbp), %eax
            leal
            2(%rax),
%edx
            movl
            -
84(%rbp), %eax
            leaq
            0(,
%rax,4), %rcx
            movq
            -
72(%rbp), %rax
            addq
            %rcx,
%rax
            movl
            %edx,
(%rax)
            movl
            -
84(%rbp), %eax
            leal
            (%rax,
%rax), %edx
            movl
            -
84(%rbp), %eax
            leaq
            0(,
%rax,4), %rcx
            movq
            -
64(%rbp), %rax
            addq
            %rcx,
%rax
            movl
            %edx,
(%rax)
            addl
            $1, -
84(%rbp)
.L6:
            movl
            -
84(%rbp), %eax
            cmpl
            -
76(%rbp), %eax
            jb
            .L7
            leaq
            -
32(%rbp), %r8
            leaq
            -
48(%rbp), %rdi
            movl
            -
76(%rbp), %ecx
            movl
            -
80(%rbp), %edx
            movq
            -
64(%rbp), %rsi
            jbe
            .L13
            leal
            2(%rax),
%edx
            movl
            %edx,
0(%rbp,%rax,4)
            leal
            (%rax,
%rax), %edx
            movl
            %edx,
(%r12,%rax,4)
            incq
            %rax
            jmp
            .L9
.L13:
            leaq
40(%rsp), %r9
            leaq
24(%rsp), %r8
            movl
            %ebx,
%ecx
            movl
            %r13d,
%edx
            movq
            %r12,
%rsi
            movq
            %rbp,
%rdi
            call
            DAXPY
            movq
48(%rsp), %rax
            subq
32(%rsp), %rax
            leaq
            .LC2(%ri
p), %rsi
            movl
            0(%rbp),
%edx
            movl
            $1, %edi
cvtsi2sdq    %rax,
%xmm0
            movq
40(%rsp), %rax
            subq
24(%rsp), %rax
cvtsi2sdq    %rax,
%xmm1
            leal
            -
1(%rbx), %eax
            movl
            0(%rbp,
%rax,4), %r8d
            movq
            %rax,
%rcx
            xorl
            %eax,
%eax
            divsd
            .LC1(%ri
p), %xmm0
            addsd
            %xmm1,
%xmm0
            movsd
            %xmm0,
8(%rsp)
            call
            jne
            .L13
.L12:
            leaq
            32(%rsp),
%r9
            leaq
            16(%rsp),
%r8
            movl
            %r14d,
%ecx
            movl
            %r12d,
%edx
            movq
            %rbp, %rsi
            movq
            %rbx, %rdi
            call
            DAXPY
            movq
            40(%rsp),
%rax
            subq
            24(%rsp),
%rax
            leaq
            .LC2(%rip)
, %rsi
            pxor
            %xmm0,
%xmm0
            movl
            (%rbx),
%edx
            pxor
            %xmm1,
%xmm1
            movl
            %r13d,
%ecx
            movl
            $1, %edi
            cvtsi2sdq
            %rax,
%xmm0
            movq
            32(%rsp),
%rax
            subq
            16(%rsp),
%rax
            cvtsi2sdq
            %rax,
%xmm1
            movl
            %r13d,
%eax
            movl
            (%rbx,
%rax,4), %r8d
            xorl
            %eax, %eax
            divsd
            .LC1(%rip)
, %xmm0
            addsd
            %xmm1,
%xmm0
            movsd
            %xmm0,
8(%rsp)
            call
__printf_chk@PLT
            movsd
            8(%rsp),
%xmm0
            leaq
            .LC3(%rip)
, %rsi
            movl
            $1, %edi
            movl
            $1, %eax
            call
(%rbx,%rdx,4)
            leal
            2(%rax),
%edx
            cmpl
            %edx,
%r14d
            jbe
            .L2
            movl
(%r12,%rdx,4), %ecx
            imull
            %r13d,
%ecx
            addl
            %ecx,
(%rbx,%rdx,4)
            leal
            3(%rax),
%edx
            cmpl
            %edx,
%r14d
            jbe
            .L2
            movl
(%r12,%rdx,4), %ecx
            imull
            %r13d,
%ecx
            addl
            %ecx,
(%rbx,%rdx,4)
            leal
            4(%rax),
%edx
            cmpl
            %edx,
%r14d
            jbe
            .L2
            movl
(%r12,%rdx,4), %ecx
            addl
            $5, %eax
            imull
            %r13d,
%ecx
            addl
            %ecx,
(%rbx,%rdx,4)
            cmpl
            %eax,
%r14d
            jbe
            .L2
            imull
(%r12,%rax,4), %r13d
            addl
            %r13d,
(%rbx,%rax,4)
.L2:
            addq
            $16, %rsp
            .cfi_reme
mber_state
            .cfi_def_
cfa_offset 48
            movq
            %rbp,
%rsi
            xorl
            %edi,
%edi
            popq
            %rbx
            .cfi_def_
cfa_offset 40
            popq
            %rbp
            .cfi_def_
cfa_offset 32
            popq
            %r12
```

```
            movq    -72(%rbp), %rax
            movq    %r8, %r9
            movq    %rdi, %r8
            movq    %rax, %rdi
            call    DAXPY
            movq    -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   .LC1(%rip), %xmm2
            divsd   %xmm2, %xmm0
            addsd   %xmm1, %xmm0
            movsd   %xmm0, -56(%rbp)
            movl    -76(%rbp), %eax
            subl    $1, %eax
            movl    %eax, %eax
            leaq    0(,%rax,4), %rdx
            movq    -72(%rbp), %rax
            addq    %rdx, %rax
            movl    (%rax), %edx
            movl    -76(%rbp), %eax
            leal    -1(%rax), %esi
            movq    -72(%rbp), %rax
            movl    (%rax), %eax
            movl    %edx, %ecx
            movl    %esi, %edx
            movl    %eax, %esi
            movl    $.LC0, %eax
            movq    %rax, %rdi
            movl    $0, %eax
            call    __printf_chk@PLT
            movsd   8(%rsp), %xmm0
            leaq    .LC3(%rip), %rsi
            movl    $1, %edi
            movb    $1, %al
            call    __printf_chk@PLT
            movq    %rbp, %rdi
            call    free@PLT
            movq    %r12, %rdi
            call    free@PLT
            xorl    %eax, %eax
            movq    56(%rsp), %rcx
            xorq    %fs:40, %rcx
            je      .L11
            call    __stack_chk_fail@PLT
.L11:
            addq    $64, %rsp
            .cfi_def_cfa_offset 48
            popq    %rbx
            .cfi_def_cfa_offset 40
            popq    %rbp
            .cfi_def_cfa_offset 32
            popq    %r12
            .cfi_def_cfa_offset 24
            popq    %r13
            .cfi_def_cfa_offset 16
            popq    %r14
            .cfi_def_cfa_offset 8
            ret
            .cfi_endproc
.LFE24:
            .size   main, .-main
            .section        .rodata.cst8,"aM",@progbits,8
            .align 8
.LC1:
            .long   0
            .long   1104006501
            .ident  "GCC: (Ubuntu 7.3.0-16ubuntu3) 7.3.0"
            .section        .note.GNU-stack,"",@progbits
            movq    %rbx, %rdi
            call    free@PLT
            movq    %rbp, %rdi
            call    free@PLT
            xorl    %eax, %eax
            movq    56(%rsp), %rcx
            xorq    %fs:40, %rcx
            jne     .L19
            addq    $64, %rsp
            .cfi_remember_state
            .cfi_def_cfa_offset 48
            popq    %rbx
            .cfi_def_cfa_offset 40
            popq    %rbp
            .cfi_def_cfa_offset 32
            popq    %r12
            .cfi_def_cfa_offset 24
            popq    %r13
            .cfi_def_cfa_offset 16
            popq    %r14
            .cfi_def_cfa_offset 8
            ret
.L18:
            .cfi_restore_state
            movq    stderr(%rip), %rcx
            leaq    .LC0(%rip), %rdi
            movl    $40, %edx
            movl    $1, %esi
            call    fwrite@PLT
            movl    $1, %edi
            call    exit@PLT
.L19:
            call    __stack_chk_fail@PLT
            .cfi_endproc
.LFE42:
            .size   main, .-main
            .section        .rodata.cst8,"aM",@progbits,8
            .align 8
.LC1:
            .long   0
            .long   1104006501
            .ident  "GCC: (Ubuntu 7.3.0-16ubuntu3) 7.3.0"
            .cfi_def_cfa_offset 24
            popq    %r13
            .cfi_def_cfa_offset 16
            popq    %r14
            .cfi_def_cfa_offset 8
            jmp     clock_gettime@PLT
            .p2align 4,,10
            .p2align 3
.L3:
            .cfi_restore_state
            leal    -1(%r14), %eax
            leaq    4(,%rax,4), %rcx
            xorl    %eax, %eax
            .p2align 4,,10
            .p2align 3
.L8:
            movl    (%r12,%rax), %edx
            imull   %r13d, %edx
            addl    %edx, (%rbx,%rax)
            addq    $4, %rax
            cmpq    %rcx, %rax
            jne     .L8
            jmp     .L2
            .cfi_endproc
.LFE41:
            .size   DAXPY, .-DAXPY
            .section        .rodata.str1.8,"aMS",@progbits,1
            .align 8
.LC0:
            .string "ERROR: falta tam del vector y constante\n"
            .section        .rodata.str1.1,"aMS",@progbits,1
.LC5:
            .string "y[0] = %i, y[%i] = %i\n"
.LC6:
            .string "\nTiempo (seg.) = %11.9f\n"
            .section        .text.startup,"ax",@progbits
            .p2align 4,,15
            .globl  main
            .type
```

```
%ecx
        movl    %esi, %edx
        movl    %eax, %esi
        leaq    .LC2(%rip), %rdi
        movl    $0, %eax
        call    printf@PLT
        movq    -56(%rbp), %rax
        movq    %rax, -120(%rbp)
        movsd   -120(%rbp), %xmm0
        leaq    .LC3(%rip), %rdi
        movl    $1, %eax
        call    printf@PLT
        movq    -72(%rbp), %rax
        movq    %rax, %rdi
        call    free@PLT
        movq    -64(%rbp), %rax
        movq    %rax, %rdi
        call    free@PLT
        movl    $0, %eax
        movq    -8(%rbp), %rcx
        xorq    %fs:40, %rcx
        je      .L9
        call    __stack_chk_fail@PLT
.L9:
        leave
        .cfi_def_cfa 7, 8
        ret
        .cfi_endproc
.LFE6:
        .size   main, .-main
        .section        .rodata
        .align 8
.LC1:
        .long   0
        .long   1104006501
        .ident  "GCC: (Ubuntu 7.3.0-16ubuntu3) 7.3.0"
        .section        .note.GNU-stack,"",@progbits
```

```
        .section        .note.GNU-stack,"",@progbits
```

```
        main, @function
main:
.LFB42:
        .cfi_startproc
        pushq   %r14
        .cfi_def_cfa_offset 16
        .cfi_offset 14, -16
        pushq   %r13
        .cfi_def_cfa_offset 24
        .cfi_offset 13, -24
        pushq   %r12
        .cfi_def_cfa_offset 32
        .cfi_offset 12, -32
        pushq   %rbp
        .cfi_def_cfa_offset 40
        .cfi_offset 6, -40
        pushq   %rbx
        .cfi_def_cfa_offset 48
        .cfi_offset 3, -48
        subq    $64, %rsp
        .cfi_def_cfa_offset 112
        movq    %fs:40, %rax
        movq    %rax, 56(%rsp)
        xorl    %eax, %eax
        cmpl    $2, %edi
        jle     .L45
        movq    16(%rsi), %rdi
        movq    %rsi, %rbp
        movl    $10, %edx
        xorl    %esi, %esi
        movl    $-1, %r14d
        call    strtol@PLT
        movq    8(%rbp), %rdi
        xorl    %esi, %esi
        movl    $10, %edx
        movq    %rax, %rbx
        call    strtol@PLT
        movl    %eax, %r12d
        movq
```

			%rax, %r13
			salq $2, %r12
			movq %r12, %rdi
			call malloc@PLT
			movq %r12, %rdi
			movq %rax, %rbp
			call malloc@PLT
			testl %r13d, %r13d
			movq %rax, %r12
			je .L30
			movq %rbp, %rdx
			addl %r13d, %r14d
			movl $4, %ecx
			shrq $2, %rdx
			negq %rdx
			andl $3, %edx
			leal 3(%rdx), %eax
			cmpl $4, %eax
			cmovb %ecx, %eax
			cmpl %eax, %r14d
			jb .L37
			testl %edx, %edx
			movl $0, 8(%rsp)
			je .L32
			cmpl $1, %edx
			movl $2, 0(%rbp)
			movl $0, (%r12)
			je .L39
			cmpl $3, %edx
			movl $3, 4(%rbp)
			movl $2, 4(%r12)
			jne .L40
			movl $4, 8(%rbp)
			movl $4, 8(%r12)

			movl $3, 8(%rsp)
			.L32:
			movd 8(%rsp), %xmm4
			movl %r13d, %r8d
			xorl %eax, %eax
			subl %edx, %r8d
			movdqa .LC2(%rip), %xmm3
			pshufd $0, %xmm4, %xmm0
			salq $2, %rdx
			movl %r8d, %edi
			movdqa .LC3(%rip), %xmm2
			leaq 0(%rbp, %rdx), %rsi
			shrl $2, %edi
			addq %r12, %rdx
			xorl %ecx, %ecx
			paddd .LC1(%rip), %xmm0
			.p2align 4,,10
			.p2align 3
			.L34:
			movdqa %xmm0, %xmm1
			addl $1, %ecx
			paddd %xmm2, %xmm1
			movaps %xmm1, (%rsi,%rax)
			movdqa %xmm0, %xmm1
			paddd %xmm3, %xmm0
			pslld $1, %xmm1
			movups %xmm1, (%rdx,%rax)
			addq $16, %rax
			cmpl %edi, %ecx
			jb .L34
			movl %r8d, %edx
			movl 8(%rsp), %eax
			andl $-4, %edx
			addl %edx,

```
%eax
                cmpl
                %edx,
%r8d
                je
                .L30
.L31:
                leal
                2(%rax),
%edx
                leal
                (%rax,
%rax), %esi
                movl
                %eax,
%ecx
                movl
                %edx,
0(%rbp,%rcx,4)
                movl
                %esi,
(%r12,%rcx,4)
                leal
                1(%rax),
%ecx
                cmpl
                %ecx,
%r13d
                jbe
                .L30
                leal
                3(%rax),
%esi
                movl
                %ecx,
%edi
                addl
                %ecx,
%ecx
                cmpl
                %edx,
%r13d
                movl
                %ecx,
(%r12,%rdi,4)
                movl
                %esi,
0(%rbp,%rdi,4)
                jbe
                .L30
                leal
                4(%rax),
%ecx
                movl
                %edx,
%edi
                addl
                %edx,
%edx
                cmpl
                %esi,
%r13d
                movl
                %edx,
(%r12,%rdi,4)
                movl
                %ecx,
0(%rbp,%rdi,4)
                jbe
                .L30
                leal
                5(%rax),
%edx
                movl
                %esi,
%edi
                addl
                %esi,
%esi
                cmpl
                %ecx,
%r13d
                movl
                %esi,
(%r12,%rdi,4)
                movl
                %edx,
0(%rbp,%rdi,4)
```

			jbe .L30
			leal 6(%rax), %edi
			movl %ecx, %esi
			addl %ecx, %ecx
			cmpl %edx, %r13d
			movl %ecx, (%r12,%rsi,4)
			movl %edi, 0(%rbp,%rsi,4)
			jbe .L30
			movl %edx, %ecx
			addl $7, %eax
			addl %edx, %edx
			movl %eax, 0(%rbp,%rcx,4)
			movl %edx, (%r12,%rcx,4)
			.L30:
			leaq 32(%rsp), %r9
			leaq 16(%rsp), %r8
			movl %ebx, %edx
			movl %r13d, %ecx
			movq %r12, %rsi
			movq %rbp, %rdi
			call DAXPY
			movq 40(%rsp), %rax
			subq 24(%rsp), %rax
			leaq .LC5(%rip), %rsi
			pxor %xmm0, %xmm0
			movl 0(%rbp), %edx
			pxor %xmm1, %xmm1
			movl %r14d, %ecx
			movl $1, %edi
			cvtsi2sdq %rax, %xmm0
			movq 32(%rsp), %rax
			subq 16(%rsp),

```
%rax
                cvtsi2sdq
                %rax,
%xmm1
                movl
                %r14d,
%eax
                movl
                0(%rbp,
%rax,4), %r8d
                xorl
                %eax,
%eax
                divsd
                .LC4(%rip
), %xmm0
                addsd
                %xmm1,
%xmm0
                movsd
                %xmm0,
8(%rsp)
                call

__printf_chk@PLT
                movsd
                8(%rsp),
%xmm0
                leaq
                .LC6(%rip
), %rsi
                movl
                $1, %edi
                movl
                $1, %eax
                call

__printf_chk@PLT
                movq
                %rbp,
%rdi
                call
                free@PLT
                movq
                %r12,
%rdi
                call
                free@PLT
                xorl
                %eax,
%eax
                movq
                56(%rsp),
%rbx
                xorq
                %fs:40,
%rbx
                jne
                .L46
                addq
                $64, %rsp
                .cfi_reme
mber_state
                .cfi_def_
cfa_offset 48
                popq
                %rbx
                .cfi_def_
cfa_offset 40
                popq
                %rbp
                .cfi_def_
cfa_offset 32
                popq
                %r12
                .cfi_def_
cfa_offset 24
                popq
                %r13
                .cfi_def_
cfa_offset 16
                popq
                %r14
                .cfi_def_
cfa_offset 8
                ret
.L39:
                .cfi_rest
```

			ore_state
			movl $1, 8(%rsp)
			jmp .L32
			.L40:
			movl $2, 8(%rsp)
			jmp .L32
			.L37:
			xorl %eax, %eax
			jmp .L31
			.L45:
			movq stderr(%rip), %rcx
			leaq .LC0(%rip), %rdi
			movl $40, %edx
			movl $1, %esi
			call fwrite@PLT
			movl $1, %edi
			call exit@PLT
			.L46:
			call __stack_chk_fail@PLT
			.cfi_endproc
			.LFE42:
			.size main, .-main
			.section .rodata.cst16,"aM",@progbits,16
			.align 16
			.LC1:
			.long 0
			.long 1
			.long 2
			.long 3
			.align 16
			.LC2:
			.long 4
			.long 4
			.long 4
			.long 4
			.align 16
			.LC3:
			.long 2
			.long 2
			.long 2
			.long 2
			.section .rodata.cst8,"aM",@progbits,8
			.align 8
			.LC4:
			.long 0
			.long

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