

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:

PS1="[Fco Javier Merchan Martin \u@\h:\w] \D{%F %A}\n\$"

Denominación de marca del chip de procesamiento o procesador (se encuentra en /proc/cpuinfo):
Intel(R) Core(TM) i5-2400 CPU @ 3.10GHz

Sistema operativo utilizado: *Ubuntu 16.04.2 LTS*

Versión de gcc utilizada: *5.4.0*

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

```
LXTerminal
[FCo Javier Merchan Martin franmerchan6@i141084:/] 2018-05-22 martes
lscpu
Arquitectura:          x86_64
modo(s) de operación de las CPUs:32-bit, 64-bit
Orden de bytes:       Little Endian
CPU(s):               4
On-line CPU(s) list:  0-3
Hilo(s) de procesamiento por núcleo:1
Núcleo(s) por «socket»:4
Socket(s):             1
Modo(s) NUMA:         1
ID de fabricante:     GenuineIntel
Familia de CPU:        6
Modelo:                42
Model name:            Intel(R) Core(TM) i5-2400 CPU @ 3.10GHz
Revisión:              7
CPU MHz:               1599.951
CPU max MHz:           3400.000
CPU min MHz:           1600.000
BogoMIPS:              6219.87
Virtualización:        VT-x
Caché L1d:             32K
Caché L1i:             32K
Caché L2:              256K
Caché L3:              6144K
NUMA node0 CPU(s):    0-3
Flags:                 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 tm pbe syscall nx rdtscp lm consta
nt tsc arch_perfmon pebs bts rep good nopl xtopology nonstop tsc_aperfperf eagerfpu pni pclmulqdq dtes64 monitor ds_cpl vmx smx est tm2 ssse3 cx16 xtpr pdcm pcid sse4.1
sse4.2 x2apic popcnt tsc_deadline_timer aes xsave avx tshf lm epb tpr_shadow vnmi flexpriority ept vpid xsaveopt dtherm ida arat pln pts
[FCo Javier Merchan Martin franmerchan6@i141084:/] 2018-05-22 martes
```

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

```

Abrir  [icon] pmm-secuencial.c
PEN AZUL /media/franmerchan6/PEN AZUL/AC/Practica 4 Guardar - [icon] x

Archivo Editar Ver Buscar Herramientas Documentos Ayuda

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

#define MAX 2255 //2 elevado a 25
int a[MAX][MAX], b[MAX][MAX], c[MAX][MAX];

int main(int argc, char **argv){
    int i, j, k;

    if(argc < 2){
        fprintf(stderr, "Falta size\n");
        exit(-1);
    }

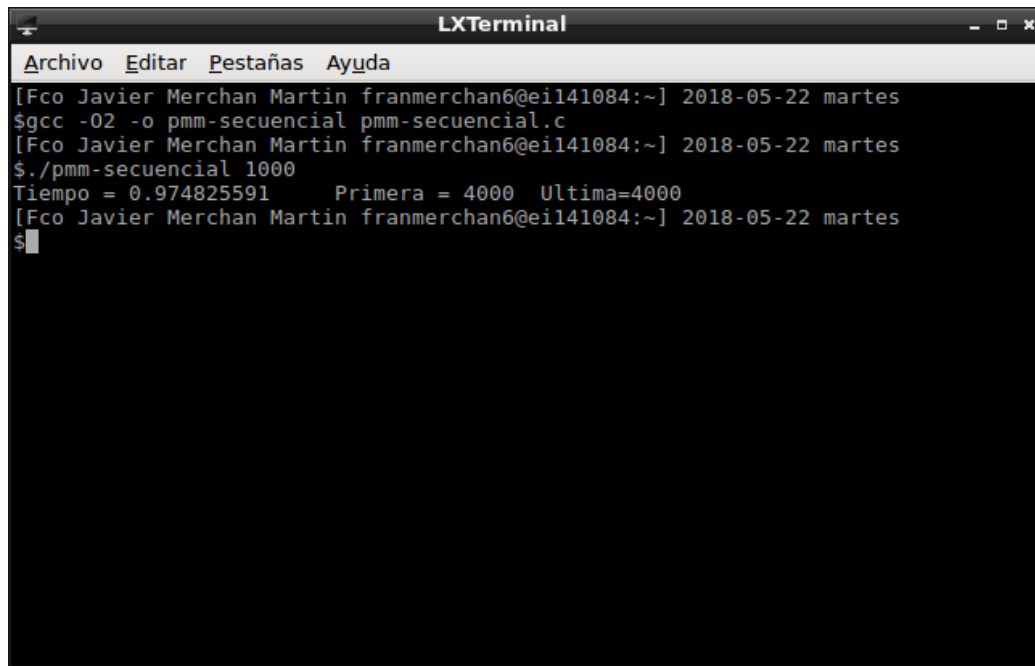
    unsigned int N = atoi(argv[1]); // Máximo N = 2^32-1=4294967295 (sizeof(unsigned int) = 4 B)

    if(N>MAX)
        N=MAX;
    // Inicializamos las matrices
    for (i=0; i<N; i++){
        for (j=0; j<N; j++){
            a[i][j] = 0;
            b[i][j] = 2;
            c[i][j] = 2;
        }
    }

    struct timespec cgt1,cgt2; double ncgt;
    clock_gettime(CLOCK_REALTIME,&cgt1);
    // Multiplicacion
    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));
    // Pitamos la primera y la ultima línea de la matriz resultante
    printf("Tiempo = %11.9f\t Primera = %d\t Última=%d\n",ncgt,a[0][0],a[N-1][N-1]);
}
C Anchura de la pestaña: 8 Ln 1, Col 1 INS

```



```
LXTerminal
Archivo Editar Pestañas Ayuda
[Fco Javier Merchan Martin franmerchan6@ei141084:~] 2018-05-22 martes
$gcc -O2 -o pmm-secuencial pmm-secuencial.c
[Fco Javier Merchan Martin franmerchan6@ei141084:~] 2018-05-22 martes
$./pmm-secuencial 1000
Tiempo = 0.974825591      Primera = 4000  Ultima=4000
[Fco Javier Merchan Martin franmerchan6@ei141084:~] 2018-05-22 martes
$
```

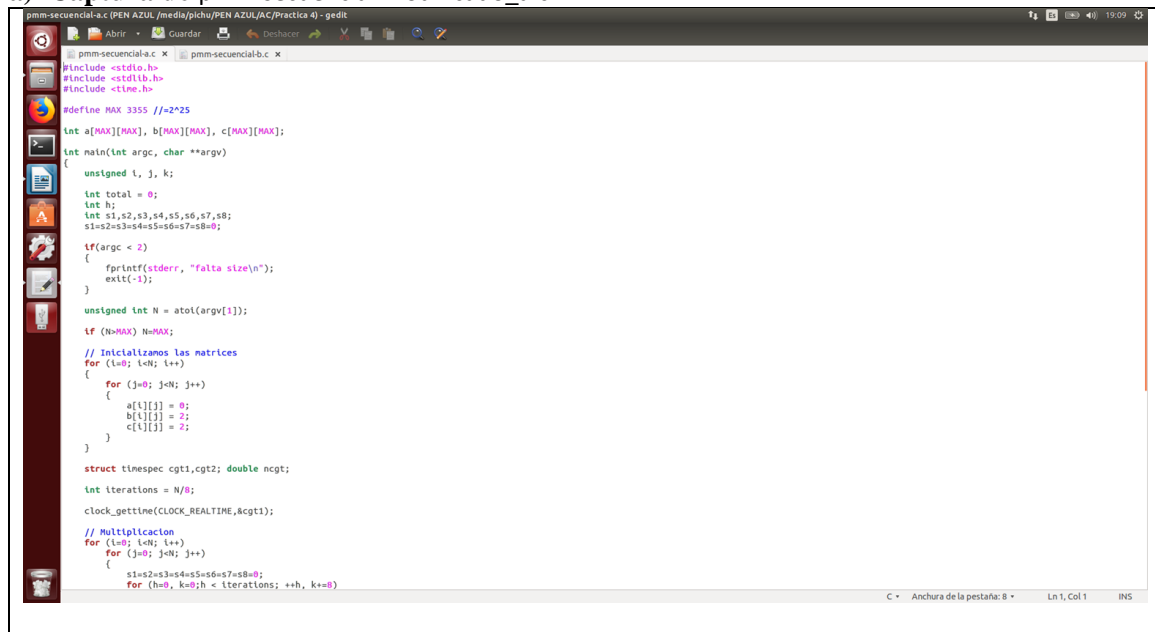
1.1. MODIFICACIONES REALIZADAS (al menos dos modificaciones):

Modificación a) –explicación-: He desenrollado el bucle k en bloques de 8

Modificación b) –explicación-: He invertido los bucles j y k, al estar así mas proximos los datos en memoria

1.1. CÓDIGOS FUENTE MODIFICACIONES

a) Captura de pmm-secuencial-modificado_a.c



```
pmm-secuencial-a.c (PEN AZUL /media/pichu/PEN AZUL/AC/Practica 4) - gedit
#include <stdio.h>
#include <stdlib.h>
#include <time.h>

#define MAX 3355 //2^25

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

int main(int argc, char **argv)
{
    unsigned i, j, k;

    int total = 0;
    int h;
    int s1,s2,s3,s4,s5,s6,s7,s8;
    s1=s2=s3=s4=s5=s6=s7=s8=0;

    if(argc < 2)
    {
        fprintf(stderr, "falta size\n");
        exit(-1);
    }

    unsigned int N = atoi(argv[1]);

    if (N>MAX) N=MAX;

    // Inicializamos las matrices
    for (i=0; i<N; i++)
    {
        for (j=0; j<N; j++)
        {
            a[i][j] = 0;
            b[i][j] = 2;
            c[i][j] = 2;
        }
    }

    struct timespec cgt1,cgt2; double ncgt;

    int iterations = N/8;

    clock_gettime(CLOCK_REALTIME,&cgt1);

    // Multiplicacion
    for (i=0; i<N; i++)
    {
        for (j=0; j<N; j++)
        {
            s1=s2=s3=s4=s5=s6=s7=s8=0;
            for (h=0, k=0; h < iterations; ++h, k+=8)
            {

```

```

pmm-secuencial-a.c x pmm-secuencial-b.c x
for (j=0; j<N; j++)
{
    a[l][j] = 0;
    b[l][j] = 2;
    c[l][j] = 2;
}

struct timespec cgt1,cgt2; double ncgt;
int iterations = N/8;
clock_gettime(CLOCK_REALTIME,&cgt1);

// Multiplicacion
for (l=0; l<N; l++)
for (j=0; j<N; j++)
{
    s1=s2=s3=s4=s5=s6=s7=s8=0;
    for (h=0; h < iterations; ++h, k+=8)
    {
        s1 += (b[l][k] * c[k][j]);
        s2 += (b[l][k+1] * c[k+1][j]);
        s3 += (b[l][k+2] * c[k+2][j]);
        s4 += (b[l][k+3] * c[k+3][j]);
        s5 += (b[l][k+4] * c[k+4][j]);
        s6 += (b[l][k+5] * c[k+5][j]);
        s7 += (b[l][k+6] * c[k+6][j]);
        s8 += (b[l][k+7] * c[k+7][j]);
    }

    total = s1 + s2 + s3 + s4 + s5 + s6 + s7 + s8;
    a[l][j]=total;

    for(k=iterations*8; k<N; ++k)
        total += (b[l][k]*c[k][j]);

    a[l][j]=total;
}

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

// Pitanos la primera y la ultima línea de la matriz resultante
printf("Tiempo = %11.9f\t Primera = %d\t Ultima=%d\n",ncgt,a[0][0],a[N-1][N-1]);

return 0;

```

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

```

LXTerminal
Archivo Editar Pestañas Ayuda
[FCo Javier Merchan Martin franmerchan6@eil41084:~] 2018-05-22 martes
$gcc -O2 -o pmm-secuencial pmm-secuencial-a.c
[FCo Javier Merchan Martin franmerchan6@eil41084:~] 2018-05-22 martes
$./
.cache/          Documentos/      .local/          pmm-secuencial
.config/         Escritorio/      .mozilla/        Público/
.dbus/           .face           Música/          .thumbnails/
Descargas/       Imágenes/       Plantillas/      Vídeos/
[FCo Javier Merchan Martin franmerchan6@eil41084:~] 2018-05-22 martes
$./pmm-secuencial 1000
Tiempo = 0.513031488      Primera = 4000  Ultima=4000
[FCo Javier Merchan Martin franmerchan6@eil41084:~] 2018-05-22 martes
$

```

b) Captura de pmm-secuencial-modificado_b.c

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

#define MAX 3355 //2^25

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

int main(int argc, char **argv)
{
    unsigned i, j, k;

    if(argc < 2)
    {
        fprintf(stderr, "falta size(n)\n");
        exit(-1);
    }

    unsigned int N = atoi(argv[1]);

    if (N>MAX) N=MAX;

    // Inicializamos las matrices
    for (i=0; i<N; i++)
    {
        for (j=0; j<N; j++)
        {
            a[i][j] = 0;
            b[i][j] = 2;
            c[i][j] = 2;
        }
    }

    struct timespec cgt1,cgt2; double ncgt;

    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][j] += b[i][k] * c[k][j];

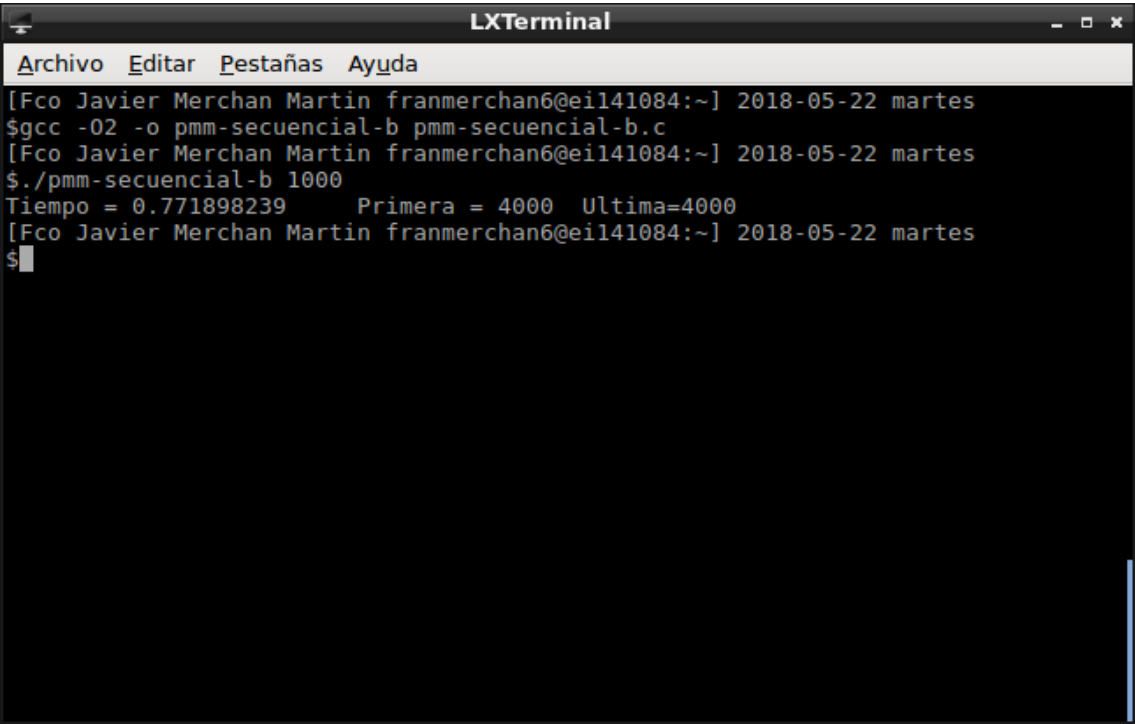
    clock_gettime(CLOCK_REALTIME,&cgt2);

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

    // Muestramos la primera y la ultima linea de la matriz resultante
    printf("Tiempo = %s1.9f\t Primera = %d\t Ultima=%d\n",ncgt,a[0][0],a[N-1][N-1]);

    return 0;
}
```

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



1.1. TIEMPOS:

Modificación	-O2
Sin modificar	0.974825591
Modificación a)	0.513031488
Modificación b)	0.771898239
...	

1.1. COMENTARIOS SOBRE LOS RESULTADOS: Ha habido mejor sobre todo la modificación a, la cual a consistido en hacer los bucles en bloques de 8 en 8 en vez de 1 en 1, reduciendo así el número de saltos y de iteraciones con la contrapartida de hacer el código más grande y que esas iteraciones se han algo más largas, debido al sobrecálculo en cada iteración con respecto al bucle inicial.

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

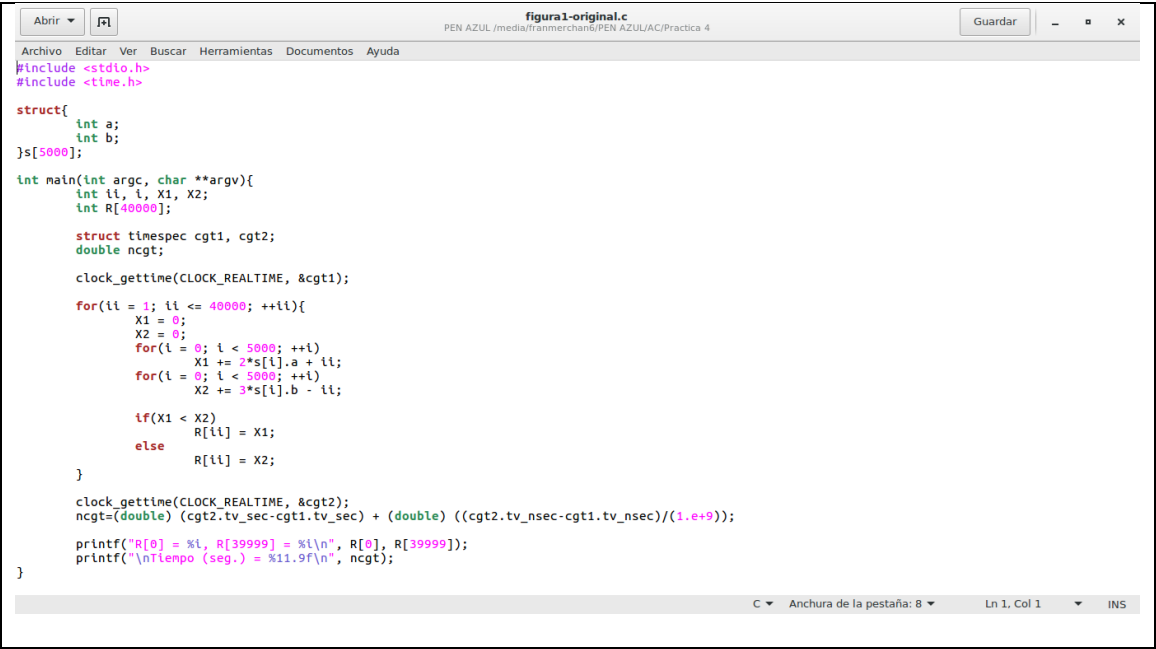
pmm-secuencial.s	pmm-secuencial-modificado_b.s	pmm-secuencial-modificado_c.s
<pre> movq %r13, %rax movl \$b, %r9d addq \$c, %rbp imulq \$-9020, %r13, %r13 negq %rax imulq \$9024, %r14, %r11 leaq -9020(%r13), %r10 leaq -4(%rax,4), %r13 addq \$a, %r11 .L8: leaq 0(%r13,%r12), %r8 movq %rbp, %rdi .p2align 4,,10 .p2align 3 .L12: movl (%r8), %esi leaq (%r10,%rdi), %rax movq %r9, %rcx .p2align 4,,10 .p2align 3 .L9: movl (%rcx), %edx addq \$9020, %rax addq \$4, %rcx imull -9020(%rax), %edx addl %edx, %esi cmpq %rdi, %rax jne .L9 movl %esi, (%r8) addq \$4, %r8 leaq 4(%rax), %rdi </pre>	<pre> movl %ebx, %eax leal 0(%rbx,8), %ebx movq \$a, 64(%rsp) subl \$1, %eax movq \$b, 40(%rsp) movl \$0, 72(%rsp) salq \$5, %rax movl %ebx, 52(%rsp) addq \$32, %rax movq %rax, 56(%rsp) .L14: movl 72(%rsp), %r14d movq 64(%rsp), %rax movq \$c, 32(%rsp) movq \$c+4, 24(%rsp) movl \$0, 12(%rsp) movq %rax, 16(%rsp) imulq \$3355, %r14, %r14 .p2align 4,,10 .p2align 3 .L10: movl 48(%rsp), %eax testl %eax, %eax je .L32 movq 24(%rsp), %rax movq 56(%rsp), %r13 xorl %ebp, %ebp movq 32(%rsp), %rdi xorl %ecx, %ecx xorl </pre>	<pre> xorl %edx, %edx .L10: leaq b(%rdx), %r8 xorl %esi, %esi .p2align 4,,10 .p2align 3 .L8: movl (%r8), %edi xorl %eax, %eax .p2align 4,,10 .p2align 3 .L7: movl c(%rsi,%rax), %ecx imull %edi, %ecx addl %ecx, a(%rdx,%rax) addq \$4, %rax cmpq %rbx, %rax jne .L7 addq \$13420, %rsi addq \$4, %r8 cmpq %r12, %rsi jne .L8 addq \$13420, %rdx cmpq %r12, %rdx jne .L10 .L11: leaq 16(%rsp), %rsi xorl %edi, %edi </pre>

<pre> cmpq %r12, %r8 jne .L12 addq \$9020, %r12 addq \$9020, %r9 cmpq %r11, %r12 jne .L8 .L11: leaq 16(%rsp), %rsi xorl %edi, %edi </pre>	<pre> %r8d, %r8d xorl %r9d, %r9d xorl %r10d, %r10d xorl %r11d, %r11d addq %rax, %r13 movq %rax, %rdx movq 40(%rsp), %rax xorl %ebx, %ebx xorl %r12d, %r12d .p2align 4,,10 .p2align 3 .L7: movl (%rax), %esi addq \$32, %rdx addq \$32, %rax imull (%rdi), %esi addq \$107360, %rdi addl %esi, %r12d movl -28(%rax), %esi imull -32(%rdx), %esi addl %esi, %ebx movl -24(%rax), %esi imull -28(%rdx), %esi addl %esi, %r11d movl -20(%rax), %esi imull -24(%rdx), %esi addl %esi, %r10d movl -16(%rax), %esi imull -20(%rdx), %esi addl %esi, %r9d movl -12(%rax), %esi imull -16(%rdx), %esi addl %esi, %r8d movl -8(%rax), %esi imull -12(%rdx), %esi addl %esi, %ecx movl -4(%rax), %esi imull -8(%rdx), %esi addl </pre>
--	--

		<pre> %esi, %ebp cmpq %rdx, %r13 jne .L7 addl %r12d, %ebx addl %r11d, %ebx addl %r10d, %ebx addl %ebx, %r9d addl %r9d, %r8d addl %r8d, %ecx addl %ecx, %ebp </pre>
	.L13:	<pre> movl 52(%rsp), %eax cmpl %eax, %r15d jbe .L8 movl 12(%rsp), %ecx movl %eax, %edx imulq \$3355, %rcx, %rcx .p2align 4,,10 .p2align 3 </pre>
	.L9:	<pre> movl %edx, %eax addl \$1, %edx leaq (%r14,%rax), %rsi addq %rcx, %rax movl b(,%rsi,4), %esi imull c(,%rax,4), %esi addl %esi, %ebp cmpl %r15d, %edx jne .L9 </pre>
	.L8:	<pre> addl \$1, 12(%rsp) movq 16(%rsp), %rax movl 12(%rsp), %ebx addq \$4, 32(%rsp) addq \$13420, 24(%rsp) movl %ebp, (%rax) addq \$4, %rax cmpl %r15d, %ebx movq %rax, 16(%rsp) jne </pre>

	<pre>.L10 addl \$1, 72(%rsp) addq \$13420, 64(%rsp) movl 72(%rsp), %eax addq \$13420, 40(%rsp) cmpl %r15d, %eax jne .L14 jmp .L15 .L3: leaq 80(%rsp), %rsi xorl %edi, %edi</pre>	
--	---	--

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



```
Abrir  figura1-original.c
PEN AZUL /media/franmerchan6/PEN AZUL/AC/Practica 4  Guardar  -  +  x

Archivo  Editar  Ver  Buscar  Herramientas  Documentos  Ayuda

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

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

int main(int argc, char **argv){
    int i1, i, X1, X2;
    int R[40000];

    struct timespec cgt1, cgt2;
    double ncgt;

    clock_gettime(CLOCK_REALTIME, &cgt1);

    for(i1 = 1; i1 <= 40000; ++i1){
        X1 = 0;
        X2 = 0;
        for(i = 0; i < 5000; ++i)
            X1 += 2*s[i].a + i1;
        for(i = 0; i < 5000; ++i)
            X2 += 3*s[i].b - i1;

        if(X1 < X2)
            R[i1] = X1;
        else
            R[i1] = 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);
}
```

```

LXTerminal
Archivo  Editar  Pestañas  Ayuda
[FCo Javier Merchan Martin franmerchan6@eil43089:~] 2018-05-29 martes
$gcc -O2 -o figural-original figural-original.c
[FCo Javier Merchan Martin franmerchan6@eil43089:~] 2018-05-29 martes
$./figural-original
R[0] = 0, R[39999] = -199995000

Tiempo (seg.) = 0.297566603
[FCo Javier Merchan Martin franmerchan6@eil43089:~] 2018-05-29 martes
$

```

1.1. MODIFICACIONES REALIZADAS (al menos dos modificaciones):

Modificación a) –explicación–: poner a y b 5000 veces en vez de del struc

Modificación b) –explicación–: cambiar el if el se por otra sentencia mas optima

1.1. CÓDIGOS FUENTE MODIFICACIONES

a) Captura figural-modificado_a.c

```

figural-modificado_a.c (PEN AZUL /media/pichu/PEN AZUL/AC/Practica 4) - gedit
#include <stdio.h>
#include <time.h>

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

int main(int argc, char **argv){
    int i, l, X1, X2;
    int R[40000];

    struct timespec cgt1, cgt2;
    double ncgt;

    clock_gettime(CLOCK_REALTIME, &cgt1);

    for(l = 1; l <= 40000; ++l){
        X1 = 0;
        X2 = 0;
        for(i = 0; i < 5000; ++i)
            X1 += 2*a[i] + i;
        for(i = 0; i < 5000; ++i)
            X2 += 3*b[i] - i;

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

```

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

```

LXTerminal
Archivo Editar Pestañas Ayuda
[FCo Javier Merchan Martin franmerchan6@e1143089:~] 2018-05-29 martes
$gcc -O2 -o figur1-modificadoa figur1-modificadoa.c
[FCo Javier Merchan Martin franmerchan6@e1143089:~] 2018-05-29 martes
$./figur1-modificadoa
R[0] = 0, R[39999] = -199995000

Tiempo (seg.) = 0.299107575
[FCo Javier Merchan Martin franmerchan6@e1143089:~] 2018-05-29 martes
$

```

b) Captura figur1-modificado_b.c

```

figur1-modificado_b.c (PEN AZUL /media/pichu/PEN AZUL/AC/Practica 4) - gedit
# include <stdio.h>
# include <time.h>

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

int main(int argc, char **argv){
    int i, l, X1, X2;
    int R[40000];

    struct timespec cgt1, cgt2;
    double ncgt;

    clock_gettime(CLOCK_REALTIME, &cgt1);

    for(i = 1; i <= 40000; ++i){
        X1 = 0;
        X2 = 0;
        for(l = 0; l < 5000; ++l){
            X1 += 2*s[l].a + i;
            for(l = 0; l < 5000; ++l){
                X2 += 3*s[l].b - i;
            }
            R[i] = (X1 < X2) ? X1 : X2;
        }

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

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

```

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

```
LXTerminal
Archivo Editar Pestañas Ayuda
[FCo Javier Merchan Martin franmerchan6@i143089:~] 2018-05-29 martes
$gcc -O2 -o figural-modiciadob figural-modiciadob.c
figural-modiciadob.c: In function 'main':
figural-modiciadob.c:26:9: warning: iteration 39999u invokes undefined behavior [-Waggressive-loop-optimizations]
   R[i1] = (X1 < X2) ? X1 : X2;
   ^
figural-modiciadob.c:18:2: note: containing loop
   for(i1 = 1; i1 <= 40000; ++i1){
   ^
[FCo Javier Merchan Martin franmerchan6@i143089:~] 2018-05-29 martes
$./figural-modiciadob
R[0] = 0, R[39999] = -199995000
Tiempo (seg.) = 0.295126233
[FCo Javier Merchan Martin franmerchan6@i143089:~] 2018-05-29 martes
$
```

1.1. TIEMPOS:

Modificación	-O2
Sin modificar	0,297566603
Modificación a)	0,299107575
Modificación b)	0,2995126233
...	

1.1. COMENTARIOS SOBRE LOS RESULTADOS: Como podemos observar ninguna de las opciones mejora el tiempo de ejecucion.

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

pmm-secuencial.s	pmm-secuencial-modificado_b.s	pmm-secuencial-modificado_c.s
<pre>.L2: leaq 52(%rsp), %r9 movl \$1, %ecx movl \$s+40004, %r8d .p2align 4,,10 .p2align 3 movl \$s, %eax xorl %esi, %esi .p2align 4,,10 .p2align 3 .L3: movl (%rax), %edx addq \$8, %rax</pre>	<pre>.L2: leaq 52(%rsp), %r9 movl \$1, %ecx movl \$s+40000, %r8d .p2align 4,,10 .p2align 3 movl \$s, %edx movl \$s+20000, %eax xorl %esi, %esi .p2align 4,,10 .p2align 3 .L3: movl (%rdx), %edi</pre>	<pre> leaq 52(%rsp), %r10 movl \$1, %esi movl \$s+40000, %r9d movl \$s+40004, %r8d .p2align 4,,10 .p2align 3 .L2: movl \$s, %eax xorl %ecx, %ecx .p2align 4,,10 .p2align 3 .L3: movl (%rax), %edx</pre>

.L4:	<pre> leal (%rcx,%rdx,2), %edx addl %edx, %esi cmpq \$5+40000, %rax jne .L3 movl \$5+4, %eax xorl %edi, %edi .p2align 4,,10 .p2align 3 movl (%rax), %edx addq \$8, %rax leal (%rdx,%rdx,2), %edx subl %ecx, %edx addl %edx, %edi cmpq %rax, %r8 jne .L4 cmpl %edi, %esi jge .L5 movl %esi, (%r9) </pre>	.L4:	<pre> addq \$4, %rdx leal (%rcx,%rdi,2), %edi addl %edi, %esi cmpq \$5+20000, %rdx jne .L3 xorl %edi, %edi .p2align 4,,10 .p2align 3 movl (%rax), %edx addq \$4, %rax leal (%rdx,%rdx,2), %edx subl %ecx, %edx addl %edx, %edi cmpq %rax, %r8 jne .L4 cmpl %edi, %esi jge .L5 movl %esi, (%r9) </pre>	.L4:	<pre> addq \$8, %rax leal (%rsi,%rdx,2), %edx addl %edx, %ecx cmpq %rax, %r9 jne .L3 movl \$5+4, %eax xorl %edi, %edi .p2align 4,,10 .p2align 3 movl (%rax), %edx addq \$8, %rax leal (%rdx,%rdx,2), %edx subl %esi, %edx addl %edx, %edi cmpq %rax, %r8 jne .L4 cmpl %edi, %ecx cmovg %edi, %ecx addl \$1, %esi addq \$4, %r10 movl %ecx, -4(%r10) cmpl \$40001, %esi jne .L2 leaq 32(%rsp), %rsi xorl %edi, %edi </pre>
	<pre> .L6: addl \$1, %ecx addq \$4, %r9 cmpl \$40001, %ecx jne .L2 leaq 32(%rsp), %rsi xorl %edi, %edi </pre>		<pre> .L6: addl \$1, %ecx addq \$4, %r9 cmpl \$40001, %ecx jne .L2 leaq 32(%rsp), %rsi xorl %edi, %edi </pre>		<pre> addl \$1, %esi addq \$4, %r10 movl %ecx, -4(%r10) cmpl \$40001, %esi jne .L2 leaq 32(%rsp), %rsi xorl %edi, %edi </pre>

repetir!!!!!!!

2. El benchmark Linpack ha sido uno de los programas más ampliamente utilizados para evaluar las prestaciones de los computadores. De hecho, se utiliza como base en la lista de los 500 computadores más rápidos del mundo (el Top500 Report). El núcleo de este programa es una rutina denominada DAXPY (*Double precision- real Alpha X Plus Y*) que multiplica un vector por una constante y los suma a otro vector (Lección 3/Tema 1):

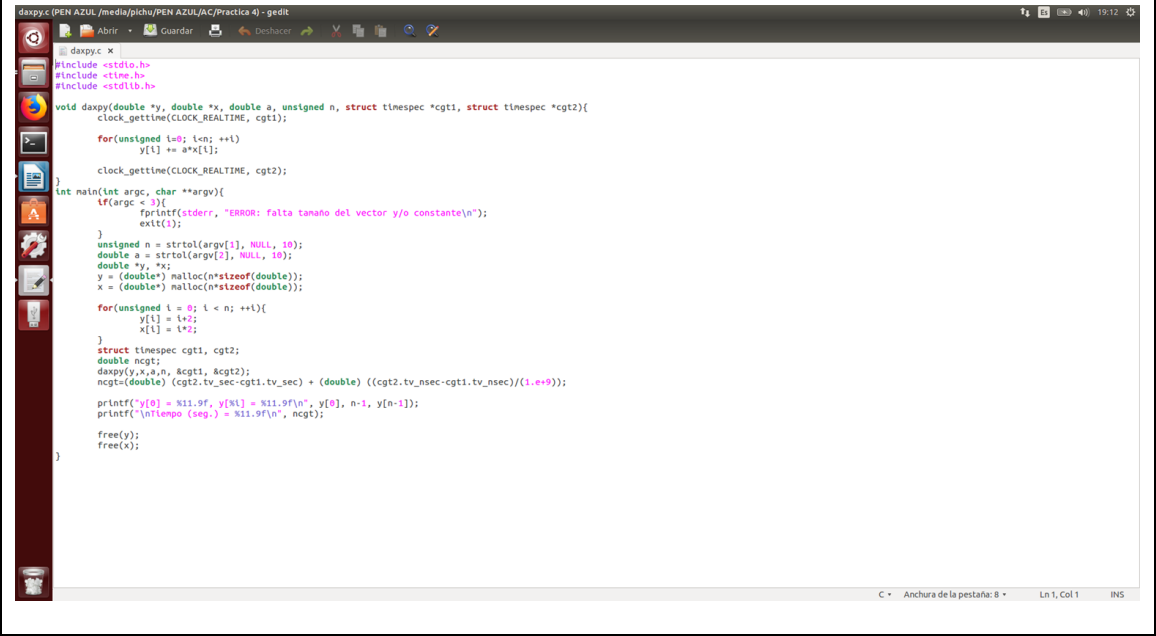
for (i=1;i<=N,i++) y[i]= a*x[i] + y[i];

2.1. Genere los programas en ensamblador para cada una de las siguientes opciones de optimización del compilador: -O0, -Os, -O2, -O3. Explique las diferencias que se observan en el código justificando al mismo tiempo las mejoras en velocidad que acarrearán. Incorpore los códigos al cuaderno de prácticas y destaque las diferencias entre ellos.

2.2. (Ejercicio EXTRA) Para la mejor de las opciones, obtenga los tiempos de ejecución con distintos valores de N y determine para su sistema los valores de Rmax (valor

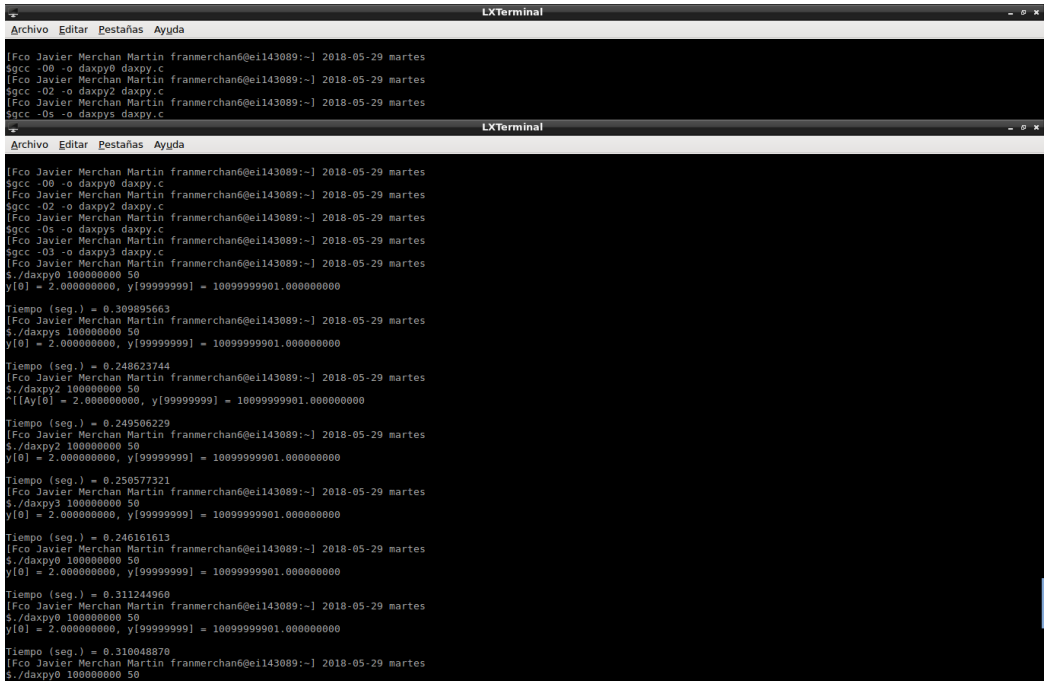
máximo del número de operaciones en coma flotante por unidad de tiempo), N_{max} (valor de N para el que se consigue R_{max}), y $N_{1/2}$ (valor de N para el que se obtiene $R_{max}/2$). Estime el valor de la velocidad pico (R_{pico}) del procesador (consulte en [4] el número de ciclos por instrucción punto flotante para la familia y modelo de procesador que está utilizando) y compárela con el valor obtenido para R_{max} . -Consulte la Lección 3 del Tema 1.

CAPTURA CÓDIGO FUENTE: daxpy.c



	-O0	-Os	-O2	-O3
Tiempos ejec.	0,000011 450	0,000004 052	0,000003 010	0,000001 852

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



COMENTARIOS QUE EXPLIQUEN LAS DIFERENCIAS EN ENSAMBLADOR:

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

daxpyO0.s	daxpyOs.s	daxpyO2.s	daxpyO3.s
<pre> movl \$0, -4(%rbp) jmp .L2 .L3: movl -4(%rbp), %eax leaq 0(%rax,4), %rdi movq - 24(%rbp), %rax addq %rax, %rdx movl -4(%rbp), %eax leaq 0(%rax,4), %rcx movq - 24(%rbp), %rax addq %rcx, %rax movl (%rax), %ecx movl -4(%rbp), %eax leaq 0(%rax,4), %rsi movq - 32(%rbp), %rax addq %rsi, %rax movl (%rax), %eax imull - 36(%rbp), %eax addl %ecx, %eax movl %eax, (%rdx) addl \$1, -4(%rbp) .L2: movl -4(%rbp), %eax cmpl - 40(%rbp), %eax jb </pre>	<pre> 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_cfa_ offset 24 popq %r12 .cfi_def_cfa_ offset 16 popq %r13 .cfi_def_cfa_ offset 8 </pre>	<pre> xorl %eax, %eax testl %ebp, %ebp je .L4 .p2align 4,,10 .p2align 3 .L5: movl 0(%r13,%rax,4) , %esi imull %r12d, %esi addl %esi, (%rbx,%rax,4) addq \$1, %rax cmpl %eax, %ebp ja .L5 .L4: popq %rbx .cfi_def_cfa_off set 40 movq %r14, %rsi xorl %edi, %edi popq %rbp .cfi_def_cfa_off set 32 popq %r12 .cfi_def_cfa_off set 24 popq %r13 .cfi_def_cfa_off set 16 popq %r14 .cfi_def_cfa_off set 8 </pre>	<pre> testl %r14d, %r14d je .L10 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, %rax andl \$15, %eax shrq \$2, %rax negq %rax andl \$3, %eax cmpl %r14d, %eax cmova %r14, %rax xorl %edx, %edx testl %eax, %eax je .L4 movl (%r12), %edx imull </pre>

<div>56(%rbp), %rax</div>	<div>.L3 movq - movq %rax, %rsi movl \$0, %edi</div>		<div>%r13d, %edx addl %edx, (%rbx) cmpl \$1, %eax movl \$1, %edx je .L4 movl 4(%r12), %ed x imull %r13d, %edx addl %edx, 4(%rbx) cmpl \$3, %eax movl \$2, %edx jne .L4 movl 8(%r12), %ed x imull %r13d, %edx addl %edx, 8(%rbx) movl \$3, %edx .L4: movl %r14d, %edi movl %r13d, 12(%rsp) xorl %ecx, %ecx subl %eax, %edi movd 12(%rsp), %x mm4 salq \$2, %rax leal -4(%rdi), %esi leaq (%rbx,%rax), %r10 xorl %r9d, %r9d pshufd \$0, %xmm4,</div>
---------------------------	---	--	--

			<pre> %xmm2 addq %r12, %rax shrq \$2, %esi addl \$1, %esi movdqa %xmm2, %xm </pre>
		m3	<pre> leal 0(%rsi,4), %r </pre>
		8d	<pre> psrlq \$32, %xmm3 </pre>
		.L6:	<pre> movdqu (%rax,%rcx), </pre>
		%xmm0	<pre> addl \$1, %r9d movdqa %xmm0, %xm </pre>
		m1	<pre> psrlq \$32, %xmm0 pmuludq %xmm3, %xm </pre>
		m0	<pre> pshufd \$8, %xmm0, </pre>
		%xmm0	<pre> pmuludq %xmm2, %xm </pre>
		m1	<pre> pshufd \$8, %xmm1, </pre>
		%xmm1	<pre> punpckldq %xmm0, %xm </pre>
		m1	<pre> movdqa (%r10,%rcx), </pre>
		%xmm0	<pre> paddb %xmm1, %xm </pre>
		m0	<pre> movaps %xmm0, </pre>
		(%r10,%rcx)	<pre> addq \$16, %rcx cmpl %esi, %r9d jb .L6 addl </pre>

				<pre> %r8d, %edx cmpl %r8d, %edi je .L10 movl %edx, %eax movl (%r12,%rax,4) , %ecx imull %r13d, %ecx addl %ecx, (%rbx,%rax,4) leal 1(%rdx), %ea x cmpl %eax, %r14d jbe .L10 movl (%r12,%rax,4) , %ecx addl \$2, %edx imull %r13d, %ecx addl %ecx, (%rbx,%rax,4) cmpl %edx, %r14d jbe .L10 movl %edx, %eax imull (%r12,%rax,4) , %r13d addl %r13d, (%rbx,%rax,4) .L10: addq \$16, %rsp .cfi_remembe r_state .cfi_def_cfa_ offset 48 movq %rbp, %rsi xorl %edi, %edi popq %rbx </pre>
--	--	--	--	--

					.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
--	--	--	--	--	---