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) i5-6300HQ CPU @ 2.30GHz

Sistema operativo utilizado: Linux version 4.4.0-43-Microsoft

Versión de gcc utilizada: gcc version 5.4.0

Volcado de pantalla que muestre lo que devuelve 1scpu en la máquina en la que ha tomado las medidas

- 1. Para el núcleo que se muestra en el Figura 1, y para un programa que implemente la multiplicación de matrices (use variables globales):
 - 1.1 Modifique el código C para reducir el tiempo de ejecución del mismo. Justifique los tiempos obtenidos (use -O2) a partir de la modificación realizada. Incorpore los códigos modificados en el cuaderno.
 - 1.2 Genere los códigos en ensamblador con -O2 para el original y dos códigos modificados obtenidos en el punto anterior (incluido el que supone menor tiempo de ejecución) e incorpórelos al cuaderno de prácticas. Destaque las diferencias entre ellos en el código ensamblador.
 - 1.3 (Ejercicio EXTRA) Intente mejorar los resultados obtenidos transformando el código ensamblador del programa para el que se han conseguido las mejores prestaciones de tiempo

Figura 1. Código C++ que suma dos vectores

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

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

        if (X1<X2) R[ii]=X1 else R[ii]=X2;
}
    ...</pre>
```

}

A) MULTIPLICACIÓN DE MATRICES:

CAPTURA CÓDIGO FUENTE: pmm-secuencial.c

1.1. MODIFICACIONES REALIZADAS (al menos dos modificaciones):

Modificación a) -explicación-: Cambiar los bucles j y k para mejorar accesos a memoria

Modificación b) -explicación-: Cambiamos las matrices por vectores

•••

1.1. CÓDIGOS FUENTE MODIFICACIONES

a) Captura de pmm-secuencial-modificado a.c

```
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);</pre>
```

b)Captura de pmm-secuencial-modificado_a.c

1.1. TIEMPOS:(ejecuto con tamaño 1000)

Fichero	-00	- o1	-02	-03	-oS
Sin modificar	10,65	3,42	2,00	2,02	3,41
Modificacion A	6,03	1,97	1,18	1,47	1,89
Modificacion B	5,97	1,17	1,17	1,21	1,23

1.1. COMENTARIOS SOBRE LOS RESULTADOS:

La mejora del apartado B es superiormente mejor que la A aunque con muy poca diferencia.

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_a.s	pmm-secuencial- modificado_b.s	
	<pre>call clock_gettime movl \$0, -112(%rbp) jmp .L14</pre>	call clock_gettime mov1 \$0, -112(%rbp) jmp .L14	<pre>call clock_gettime mov1 \$0, -96(%rbp) jmp .L10</pre>	
.L19:	mov1 \$0, -108(%rbp) jmp .L15	.L19: mov1 \$0, -104(%rbp) jmp .L15	.L15: mov1 \$0, -88(%rbp) jmp .L11	

.L18:		.L18:		.L14:	
. 110:	movl		movl	1.114:	movl
	\$0, -104(%rbp)		\$0, -108(%rbp)		\$0, -92(%rbp)
	jmp		jmp		jmp
	.L16		.L16		.L12
.L17:		.L17:		.L13:	
	movl		movl		movl
	-112(%rbp), %eax		-112(%rbp),		-96(%rbp),
	leaq	%eax		%eax	
	0(,%rax,8), %rdx		leaq		imull
	movq -96(%rbp), %rax	%rdx	0(,%rax,8),	%eax	-84(%rbp),
	addq	-01UX	movq	· can	movl
	%rdx, %rax		-96(%rbp), %rax		%eax, %edx
	movq		addq		movl
	(%rax), %rax		%rdx, %rax		-92(%rbp),
	movl		movq	%eax	
	-108(%rbp), %edx		(%rax), %rax		addl
	salq		movl		%edx, %eax
	\$2, %rdx	0 - 1	-108(%rbp),		movl
	addq %rdx, %rax	%edx	1		%eax, %eax
	movl		salq \$2, %rdx		leaq 0(,%rax,4),
	-112(%rbp), %edx		addq	%rdx	· (, oran, 4),
	leaq		%rdx, %rax	_	movq
	0(,%rdx,8), %rcx		movl		-64(%rbp),
	movq		-112(%rbp),	%rax	=
	-96(%rbp), %rdx	%edx			addq
	addq		leaq		%rax, %rdx
	%rcx, %rdx		0(,%rdx,8),		movl
	movq	%rcx	movq	%eax	-96(%rbp),
	(%rdx), %rdx movl		-96(%rbp), %rdx	seax	imull
	-108(%rbp), %ecx		addq		-84(%rbp),
	salq		%rcx, %rdx	%eax	
	\$2, %rcx		movq		movl
	addq		(%rdx), %rdx		%eax, %ecx
	%rcx, %rdx		movl		movl
	movl (%rdx), %ecx	%ecx	-108(%rbp),	%eax	-92(%rbp),
	movl	- Seca	salq	· can	addl
	-112(%rbp), %edx		\$2, %rcx		%ecx, %eax
	leaq		addq		movl
	0(,%rdx,8), %rsi		%rcx, %rdx		%eax, %eax
	movq		movl		leaq
	-88(%rbp), %rdx		(%rdx), %ecx		0(,%rax,4),
	addq %rsi, %rdx		movl -112(%rbp),	%rcx	movq
	movq	%edx	-112 (%IDP),		-64(%rbp),
	(%rdx), %rdx	o can	leaq	%rax	01(0125),
	movl		0(,%rdx,8),		addq
	-104(%rbp), %esi	%rsi			%rcx, %rax
	salq		movq		movl
	\$2, %rsi		-88(%rbp), %rdx		(%rax), %ecx
	addq		addq		movl
	%rsi, %rdx movl		%rsi, %rdx movq	%eax	-96(%rbp),
	(%rdx), %esi		(%rdx), %rdx	Juan	imull
	mov1		mov1		-84(%rbp),
	-104(%rbp), %edx		-104(%rbp),	%eax	- :
	leaq	%esi			movl
	0(,%rdx,8), %rdi		salq		%eax, %esi
	movq % rbp) % rdv		\$2, %rsi		movl
	-80(%rbp), %rdx addq		addq %rsi, %rdx	%eax	-88(%rbp),
	addq %rdi, %rdx		movl	vean	addl
	movq		(%rdx), %esi		%esi, %eax
	(%rdx), %rdx		movl		movl
	movl		-104(%rbp),		%eax, %eax
	-108(%rbp), %edi	%edx			leaq
	salq		leaq	° mai	0(,%rax,4),
	\$2, %rdi addq	%rdi	0(,%rdx,8),	%rsi	morra
	addq %rdi, %rdx	0141	movq		movq -72(%rbp),
	movl		-80(%rbp), %rdx	%rax	·= , ·-~ p / /
	(%rdx), %edx		addq		addq
				i .	
	imull		%rdi, %rdx		%rsi, %rax
	<pre>imull %esi, %edx addl</pre>		<pre>%rdi, %rdx movq (%rdx), %rdx</pre>		%rsi, %rax movl (%rax), %esi

	%ecx, %edx		movl		movl
	movl		-108(%rbp),		-88(%rbp),
	%edx, (%rax)	%edi		%eax	
	addl		salq		imull
.L16:	\$1, -104(%rbp)		\$2, %rdi addq	%eax	-84(%rbp),
	movl		%rdi, %rdx		movl
	-104(%rbp), %eax		movl		%eax, %edi
	cmpl		(%rdx), %edx		movl
	-100(%rbp), %eax		imull	0	-92(%rbp),
	jb .L17		%esi, %edx addl	%eax	addl
	addl		%ecx, %edx		%edi, %eax
	\$1, -108(%rbp)		movl		movl
.L15:			%edx, (%rax)		%eax, %eax
	movl		addl		leaq
	-108(%rbp), %eax cmpl	.L16:	\$1, -108(%rbp)	%rdi	0(,%rax,4),
	-100(%rbp), %eax		movl	9141	movq
	jb		-108(%rbp),		-80(%rbp),
	.L18	%eax		%rax	_
	addl		cmpl		addq
L14:	\$1, -112(%rbp)	%eax	-100(%rbp),		%rdi, %rax movl
	movl	ocd.X	jb		movi (%rax), %eax
	-112(%rbp), %eax		.L17		imull
	cmpl		addl		%esi, %eax
	-100(%rbp), %eax		\$1, -104(%rbp)		addl
	jb	.L15:	1		%ecx, %eax
	.L19 leag		movl -104(%rbp),		movl %eax, (%rdx)
	-48(%rbp), %rax	%eax	101 (01DP) /		addl
	movq		cmpl		\$1, -92(%rbp)
	%rax, %rsi		-100(%rbp),	.L12:	
	movl	%eax	-i-h		movl
	\$0, %edi call		jb .L18	%eax	-92(%rbp),
	clock gettime		addl		cmpl
			\$1, -112(%rbp)		-84(%rbp),
		.L14:		%eax	
			movl -112(%rbp)		jb .L13
		%eax	-112(%rbp),		addl
			cmpl		\$1, -88(%rbp)
			-100(%rbp),	.L11:	
		%eax	.,		movl
			jb .L19	%eax	-88(%rbp),
			leaq	ocan	cmpl
			-48(%rbp), %rax		-84(%rbp),
			movq	%eax	
			%rax, %rsi		jb
			movl \$0, %edi		.L14 addl
			call		\$1, -96(%rbp)
			clock_gettime	.L10:	,
			_		movl
					-96(%rbp),
				%eax	cmpl
					-84(%rbp),
				%eax	
					jb
					.L15
					leaq -32(%rbp),
				%rax	25 (2TDb) '
				· ·	movq
					%rax, %rsi
					movl
					\$0, %edi

B) CÓDIGO FIGURA 1:

CAPTURA CÓDIGO FUENTE: figural-original.c

```
int main(int argc, char **argv)
    int R[40000];
    struct timespec cgt1,cgt2; double ncgt;
    clock_gettime(CLOCK_REALTIME,&cgt1);
    for (ii = 1; ii <= 40000; ii++)
       X1 = 0; X2 = 0;
        if ( X1 < X2 )
    clock_gettime(CLOCK_REALTIME,&cgt2);
    ncgt=(double) (cgt2.tv_sec-cgt1.tv_sec)+( double) ((cgt2.tv_nsec-cgt1.tv_nsec)/(1.e+9));
    printf("R[0] = %i, R[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 elimina un for de los dos.

Modificación b) -explicación-: Se realizan 4 operaciones por cada pasada del for en vez de 1.

1.1. CÓDIGOS FUENTE MODIFICACIONES

a) Captura figural-modificado a.c

```
#include <stdio.h>
#include <time.h>
} s[5000];
int main(int argc, char **argv)
    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;
X2 += 3 * s[i].b - ii;
        if ( X1 < X2 )
    clock_gettime(CLOCK_REALTIME,&cgt2);
    ncgt=(double) (cgt2.tv_sec-cgt1.tv_sec)+( double) ((cgt2.tv_nsec-cgt1.tv_nsec)/(1.e+9));
    printf("R[0] = %i, R[39999] = %i\n", R[0], R[39999]);
    printf("\nTiempo (seg.) = %11.9f\n", ncgt);
    return 0;
```

```
[jose manuel perez lendinez jmplz14@DESKTOP-KGK4SOA:/mnt/d/AC/practica4/codigos] 2018-05-29 Tuesday
$g++ -o figura1-A.eje figura1-A.cpp
figura1-A.cpp:9:11: warning: anonymous type with no linkage used to declare variable '<anonymous struct's [5000]', with linkage
} s[5000];
[jose manuel perez lendinez jmplz14@DESKTOP-KGK4SOA:/mnt/d/AC/practica4/codigos] 2018-05-29 Tuesday
$./figura1-A.eje
R[0] = 0, R[39999] = -199995000

Tiempo (seg.) = 0.754245800
[jose manuel perez lendinez jmplz14@DESKTOP-KGK4SOA:/mnt/d/AC/practica4/codigos] 2018-05-29 Tuesday
$.
```

b) Captura figural-modificado_b.c

```
s[5000];
int main(int argc, char **argv)
   int R[40000];
   struct timespec cgt1,cgt2; double ncgt;
    clock_gettime(CLOCK_REALTIME,&cgt1);
           X1 += 2*s[i].a+ii;
           X2 += 3*s[i].b-ii;
           X1 += 2*s[i+3].a+ii;
           X2 += 3*s[i+3].b-ii;
   clock_gettime(CLOCK_REALTIME,&cgt2);
   ncgt=(double) (cgt2.tv_sec-cgt1.tv_sec)+( double) ((cgt2.tv_nsec-cgt1.tv_nsec)/(1.e+9));
    printf("\nTiempo (seg.) = %11.9f\n", ncgt);
```

```
[jose manuel perez lendinez jmplz14@DESKTOP-KGK4SOA:/mnt/d/AC/practica4/codigos] 2018-05-29 Tuesday $g++ -o figura1-B.eje figura1-B.cpp figura1-B.eje figura1-B.cpp with no linkage used to declare variable '<anonymous struct> s [5000]' with linkage } s[5000];

[jose manuel perez lendinez jmplz14@DESKTOP-KGK4SOA:/mnt/d/AC/practica4/codigos] 2018-05-29 Tuesday $./figura1-B.eje R[0] = 0, R[39999] = -199995000

Tiempo (seg.) = 0.762709400 [jose manuel perez lendinez jmplz14@DESKTOP-KGK4SOA:/mnt/d/AC/practica4/codigos] 2018-05-29 Tuesday $.
```

1.1. TIEMPOS:

Fichero	-00	-01	-02	-03
Sin modificar	1,20	0,18	0,00017	0,00018
Modificacion A	0,75	0,092	0,00018	0,00019
Modificacion B	0,75	0,093	0,00017	0,00019

1.1. COMENTARIOS SOBRE LOS RESULTADOS:

Como se ve apartir de -O2 no cambia mucho

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-secuen modificado		<pre>pmm-secuencial- modificado_c.s</pre>	
	call	Call clock get	time	call	clock gettime
	clock gettime		movl		movl
	movl		\$1,		\$1,
	\$1, -160072(%rbp)	-160072(%rbp)	1-7	-160072(%rbp)	1-7
.L9:	VI, 100072(01DP)	L7:		L7:	
.119.			1	• = / •	1
	cmpl		cmpl		cmpl
	\$40000,		\$40000,		\$40000,
-160072 (%rbp)		-160072(%rbp)		-160072(%rbp)	
	jg		jg		jg
	.L2		.L2		.L2
	movl		movl		movl
	\$0, -160064(%rbp)		\$O,		\$O,
	movl	-160064(%rbp)		-160064(%rbp)	
	\$0, -160060(%rbp)		movl	' ' '	movl
	movl		\$0,		\$0,
	\$0, -160068(%rbp)	-160060(%rbp)	⊤ ∨ ,	-160060(%rbp)	, ~ ,
.L4:	40' -100000 (stnb)	100000 (stpb)	movl	100000 (21Db)	movl
.14:					
	cmpl		\$0,		\$0,
	\$4999,	-160068(%rbp)		-160068 (%rbp)	
-160068(%rbp)		.L4:		.L4:	
	jg		cmpl		cmpl
	.L3		\$4999,		\$4999,
	movl -	-160068(%rbp)		-160068(%rbp)	
160068(%rbp), %	eax		jg		jg
	cltq		.T.3		. T.3
	movl		movl		movl
	s(,%rax,8), %eax		_		-160068(%rbp),
	leal	160068(%rbp),	9.00.	%eax	100000 (8100),
		100000(%IDD),		*eax	. 7.1
	(%rax,%rax), %edx		cltq		cltq
	movl -		movl		movl
160072(%rbp), %	eax		s(,%rax,8),		s(,%rax,8),
	addl	%eax		%eax	
	%edx, %eax		leal		leal
	addl		(%rax,%rax),		(%rax,%rax),
	%eax,	%edx		%edx	
-160064(%rbp)	•		movl		movl
(<u>-</u> /-	addl		_		-160072(%rbp),
	\$1, -160068(%rbp)	160072(%rbp),	%eax	%eax	/, (\-\p/)
	jmp	1000/2(0100),	addl	Jucan	addl
* 2	.L4		%edx, %eax		%edx, %eax
.L3:			addl		addl
	movl		%eax,		%eax,
	\$0, -160068(%rbp)	-160064(%rbp)		-160064(%rbp)	
.L6:			movl		movl
	cmpl		_		-160068(%rbp),
	\$4999,	160068(%rbp),	%eax	%eax	
-160068 (%rbp)			cltq		cltq
- 200 00 (01 DP)	jg		movl		movl
	.L5				
	· Tr 2	1	s+4(,		s+4(,%rax,8),

movl -	%rax,8), %edx	%edx
160068(%rbp), %eax	movl	movl
cltq	%edx, %eax	%edx, %eax
movl	addl	addl
s+4(,%rax,8), %edx	%eax, %eax	%eax, %eax
movl	addl	addl
%edx, %eax	%edx, %eax	%edx, %eax
addl	subl	subl
%eax, %eax	-	-160072(%rbp),
addl	160072(%rbp), %eax	%eax
%edx, %eax	addl	addl
subl -	%eax,	%eax,
160072(%rbp), %eax	-160060(%rbp)	-160060(%rbp)
addl	addl	addl
%eax,	\$1,	\$1,
-160060(%rbp)	-160068(%rbp)	-160068(%rbp)
addl		_
	jmp	jmp
\$1, -160068(%rbp)	.L4	.L4
jmp	.L3:	.L3:
.L6	movl	movl
.L5:		-160064(%rbp),
mov1 -	160064(%rbp), %eax	%eax
160064(%rbp), %eax	cmpl	cmpl
cmpl -	-	-160060(%rbp),
160060(%rbp), %eax	160060(%rbp), %eax	%eax
jge	jge	jge
.L7	.L5	.L5
movl -	movl	movl
160072(%rbp), %eax	-	-160072(%rbp),
cltq	160072(%rbp), %eax	%eax
movl -	cltq	cltq
160064(%rbp), %edx	movl	movl
movl	-	-160064(%rbp),
%edx,	160064(%rbp), %edx	%edx
-160016(%rbp,%rax,4)	movl	movl
jmp	%edx,	%edx,
.L8	-160016(%rbp,%rax,4)	-160016(%rbp,%rax,4)
.L7:	jmp	jmp
movl -	.L6	.L6
160072(%rbp), %eax	.L5:	.L5:
cltq	movl	movl
mov1 -	-	-160072(%rbp),
160060(%rbp), %edx	160072(%rbp), %eax	%eax
mov1	cltq	cltq
%edx,	movl	movl
-160016(%rbp,%rax,4)	_	-160060(%rbp),
.L8:	160060(%rbp), %edx	-100000(%IDP),
addl	mov1	movl
\$1, -160072(%rbp)	%edx,	%edx,
	*edx, -160016(%rbp,%rax,4)	
jmp		-160016(%rbp,%rax,4)
.L9	.L6:	.L6:
.L2:	addl	addl
leaq -	\$1,	\$1,
160032(%rbp), %rax	-160072(%rbp)	-160072(%rbp)
movq	jmp	jmp
%rax, %rsi	.L7	.L7
mov1	.L2:	.L2:
\$0, %edi	leaq	leaq
call	<u>-</u>	-160032(%rbp),
clock_gettime	160032(%rbp), %rax	%rax
	movq	movq
	%rax, %rsi	%rax, %rsi
	movl	movl
	\$0, %edi	\$0, %edi
	call	call
		clock_gettime
	clock_gettime	_

2. El benchmark Linpack ha sido uno de los programas más ampliamente utilizados para evaluar las prestaciones de los computadores. De hecho, se utiliza como base en la lista de los 500 computadores más rápidos del mundo (el Top500 Report). El núcleo de este

programa es una rutina denominada DAXPY (*Double precision- real Alpha X Plus Y*) que multiplica un vector por una constante y los suma a otro vector (Lección 3/Tema 1):

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

- 2.1. Genere los programas en ensamblador para cada una de las siguientes opciones de optimización del compilador: -O0, -Os, -O2, -O3. Explique las diferencias que se observan en el código justificando al mismo tiempo las mejoras en velocidad que acarrean. Incorpore los códigos al cuaderno de prácticas y destaque las diferencias entre ellos.
- 2.2. (Ejercicio EXTRA) Para la mejor de las opciones, obtenga los tiempos de ejecución con distintos valores de N y determine para su sistema los valores de Rmax (valor máximo del número de operaciones en coma flotante por unidad de tiempo), Nmax (valor de N para el que se consigue Rmax), y N1/2 (valor de N para el que se obtiene Rmax/2). Estime el valor de la velocidad pico (Rpico) del procesador (consulte en [4] el número de ciclos por instrucción punto flotante para la familia y modelo de procesador que está utilizando) y compárela con el valor obtenido para Rmax. -Consulte la Lección 3 del Tema 1.

CAPTURA CÓDIGO FUENTE: daxpy.c

```
#include <time.h>
#include <stdlib.h>
void daxpy(int *y, int *x, int a, unsigned n, struct timespec *cgt1, struct timespec *cgt2)
   clock_gettime(CLOCK_REALTIME,cgt1);
   unsigned 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");
   unsigned n = strtol(argv[1], NULL, 10);
    int a = strtol(argv[2], NULL, 10);
    unsigned i;
    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("\nTiempo (seg.) = %11.9f\n", ncgt);
```

Tiempos ejec.	-O0	-Os	-O2	-O3
Tiempos ejec.	0,3619	0,1358	0,1460	0,1653

N = 100000000

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

```
[jose manuel perez lendinez jmplz14@DESKTOP-KGK4SOA:/mnt/d/AC/practica4/codigos] 2018-05-29 Tuesday $g++ -o DAXPY.eje DAXPY.cpp
[jose manuel perez lendinez jmplz14@DESKTOP-KGK4SOA:/mnt/d/AC/practica4/codigos] 2018-05-29 Tuesday $./DAXPY.eje 100000000 20 y[0] = 2, y[99999999] = -194967335

Tiempo (seg.) = 0.406677400
[jose manuel perez lendinez jmplz14@DESKTOP-KGK4SOA:/mnt/d/AC/practica4/codigos] 2018-05-29 Tuesday $
```

COMENTARIOS QUE EXPLIQUEN LAS DIFERENCIAS EN ENSAMBLADOR:

En O0 se usan direcciones dke pila y con O2 registros. O3 desarrolla el bucle alargando el código 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)

daxpy00.s	daxpyOs.s	daxpy02.s	daxpy03.s
23 call clock_gettime 24 movl \$0, -4(%rbp) 25 .L3: 26 movl -4(%rbp), %eax 27 cmpl -40(%rbp), %eax 28 jnb .L2 29 movl -4(%rbp), %eax 30 leaq 0(,%rax,4), %rdx 31 movq -24(%rbp), %rax 32 addq %rax, %rdx 33 movl -4(%rbp), %eax 34 leaq 0(,%rax,4), %rcx 35 movq -24(%rbp), %rax 36 addq %rcx, %rax 37 movl (%rax), %ecx 38 movl -4(%rbp), %eax 39 leaq 0(,%rax,4), %rsi 40 movq -32(%rbp), %rax 41 addq %rsi, %rax 42 movl (%rax), %eax 43 imull -36(%rbp), %eax 44 addl %ecx, %eax 45 movl %eax, (%rdx) 46 addl \$1, -4(%rbp) 47 jmp .L3 48 .L2: 49 movq -56(%rbp), %rax 50 movq %rax, %rsi 51 movl \$0, %edi 52 call clock_gettime	call clock_gettime movq 8(%rsp), %r9 xorl %eax, %eax .L3: cmpl %eax, %r13d jbe .L2 movl 0(%rbp,%rax,4), %esi imull %r12d, %esi addl %esi, (%rbx,%rax,4) incq %rax jmp .L3 .L2: 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 jmp clock_gettime	call clock_gettime xorl %eax, %eax testl %ebp, %ebp je .L3 .p2align 4,,18 .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 .L3: popq %rbx .cfi_def_cfa_offset 48 movq %r14, %rsi xorl %edi, %edi popq %rbp .cfi_def_cfa_offset 32 popq %rbp .cfi_def_cfa_offset 24 popq %r12 .cfi_def_cfa_offset 24 popq %r13 .cfi_def_cfa_offset 16 popq %r14 .cfi_def_cfa_offset 8 jmp clock_gettime	call clock gettime testl %r14d, %r14d je .L18 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 %r13d, %edx addl %edx, (%rbx) cmpl \$1, %eax movl \$1, %eax movl \$1, %eax movl \$2, %eax imull %r13d, %edx addl %edx, 4(%rbx) cmpl \$2, %eax movl \$2, %eax movl \$2, %eax movl \$2, %eax imull %r13d, %edx addl %edx, 4(%rbx) cmpl \$2, %eax movl \$2, %eax movl \$2, %edx imull %r13d, %edx addl %edx, 8(%rbx) cmpl \$3, %edx .L4: movl \$(%r12), %edx imull %r13d, %edx addl %edx, 8(%rbx) movl \$3, %edx .L4: movl %r14d, %edi movl %r13d, 12(%rsp) xorl %eax, %eax subl %eax %edi

mod 12(freq), Kneat sale 22, Seri lead (*Cotal), Seri leage (Seris, Merca), Scribs prober 58, Kneat, Kneat sale 22, Seri moduly Kneat, Serial prick 534, Kneat, Kneat prober 534, Kneat, Kneat prober 534, Kneat, Kneat prober 534, Kneat, Kneat propried 544, Kneat, Kneat propried 544, Kneat, Kneat propried 544, Kneat, Kneat propried 544, Kneat moduly Kneat, Serial modul, Serial mod	1	
scale 52, Kraw load (Christ), Social loaq (Christ), Social loaq (Christ), Social load (C		movd 12(%rsp), %xmm4
lead (*Choth, News), North sect long (*Choth, News), North sect long (*North, North phothed St.), North sect long (*North, North phothed St.), North sect long (*North, North) lead (*North) le		
lean, (Nebu, Serat), New London, New Londo		
non-1 kred, kred patient 50, Konsal, Komm2 addig kra2, kress and 151, Keni and 152, Keni and 153, Keni and 153, Keni and 153, Keni and 154, Ke		
poturid 80, Komen, Komen and 52, Kersi and 53, Kersi monopa Komez, Kemmi lead (Kersi, Arab) pring 531, Kemi monopa Komez, Kemmi lead (Kersi, Kersi) monopa Komen pring 532, Komen pring 532, Komen pring 532, Komen pring 532, Komen pring 533, Komen pring 534, Komen monopa Komen, (Kersi) monopa Komen, mon		
ands \$12, Kern ands \$1, Kern ands \$1, Kern ands \$1, Kern project (Para, Mora), Nord project (Para, Mora), Nord project (Para, Mora), Nord and (Para, Mora), Nord modes (Para, Mora), Nord modes (Para, Nord) modes (Para, Nord) modes (Para, Nord) modes (Roma, Nord		
and: 31, Kori and: 31, Kori movedqu Koman, Kummi leal and (Krai, 4), Mard pariq 321, Kommi 17: movedqu (Gras, Arcs.), Kommi andia, 31, Krod moving (Gras, Arcs.), Kommi andia, 31, Krod moving (Gras, Arcs.), Kommi paright 32, Kommi, Kommi paright 34, Kommi, Kommi paright 34, Kommi, Kommi paright 36, Komm		
moving Names, Names leal (M. Neri, A.), Nerd peria, 252, Names 1.7: movings (Mrss, Ners), Names addi Si, Nerd movings Names, Names perila, 132, Names perila, Names moving (Ners, Names), Names moving (Ners, Names), Names moving (Ners, Names) moving Mrss, Names moving Mrss,		
need que, Kerni, 4), Nord pariq \$22, Kamal 127: needqu (fara, forca), Kamal addi \$13, Kernd moudqu Kaman, Kamal pariq \$23, Kamal pullud Kaman, Kamal pullud \$23, Kamal pullud \$24, Kamal pullud		
inai ag. 121, 20, 201 pariag. \$212, 20, 201 1.07: movedqu (firax, forca), Kamell and il 31, 1red movedqu (firax, forca), Kamell profiled \$13, Kamell prof		
puris \$12, Knume add st. Kred moving Knume, Knume add st. Kred moving Knume, Knume priling \$12, Knume priling \$12, Knume priling Knume,		
mordaya Kuman, Kuran mordaya Kuman addi Si, Krind mordaya Kuman purla Si2, Kuman purla Si2, Kuman puluda Kuman, Kuman puhuda Sia, Kuman kuman mordaya (Krin, Kuman mordaya (Krin, Kuman mordaya (Krin, Kuman mordaya (Krin, Kuman mordaya Kuman, Kuman mordaya Sia, Kuma addi Sia, Kuma mordaya Kuman, Kuman mordaya ku		leal 0(,%rsi,4), %r8d
mording (Cran, Even) Knewn and side is 1, Krewn mording Knewn, Knewn pruling \$32, Knewn pruling \$32, Knewn pruling \$33, Knewn pruling \$38, Knewn, Knewn pruling \$38, Knewn, Knewn pruling \$38, Knewn, Knewn purpleking \$38, Knewn, Knewn purpleking \$38, Knewn, Knewn purpleking \$38, Knewn, Knewn nowaya (Krila, Knewn, Knewn nowaya (Krila, Knewn, Knewn) nowaya Knewn, Krila, Knewn nowaya Knewn, Knewn, Knewn, Krila, Knewn nowaya Knewn, Kne		psrlq \$32, %xmm3
add 15, Krnd morday Xumma, Kummi purils \$12, Yumma punils \$2, Xumma punils \$4, Xumma, Xummi movidas (Xumb, Xumma, Xummi movidas (Xumb, Xuma, Xummi movidas (Xumb, Xuma, Xummi movidas (Xumb, Xuma, Xummi movidas (Xumb, Xuma, Xummi movidas State, Xuma, Xummi movidas State, Xuma movi Xums, Xummi, Xummi movidas State, Xuma movi Xums, Xummi movi Xums, Xummi movi Xums, Xums movi Xums mov		.L7:
add 15, Krnd morday Xumma, Kummi purils \$12, Yumma punils \$2, Xumma punils \$4, Xumma, Xummi movidas (Xumb, Xumma, Xummi movidas (Xumb, Xuma, Xummi movidas (Xumb, Xuma, Xummi movidas (Xumb, Xuma, Xummi movidas (Xumb, Xuma, Xummi movidas State, Xuma, Xummi movidas State, Xuma movi Xums, Xummi, Xummi movidas State, Xuma movi Xums, Xummi movi Xums, Xummi movi Xums, Xums movi Xums mov		movdqu (%rax,%rcx), %xmm0
mondaya Kamed, paral print 38 st. Keemed, remain purulung Kamen, keemed purulung di Kamen, keemed purulung di Kamen, keemed purulung di Karis, Kamed purulung di Karis, Karis and la Karis, (Karis, Karis, a) lesal 1(Karis, Karis, a) lesal 1(Ka		
purley Xxmma, Xxmma pulled Xxmma, Xxmma pulled Xxmma, Xxmma pulled Xxmma, Xxmma pupurled Sa, Xxmma, Xxmma pupurled Xxmma, Xxmma pupul Xxmia, Xxmia pupul Xxmia pupul Xxmia, Xxmia pupul X		
pmilided Named, Keemed publided Sak, Keemed, Keemed pmilided Named, Keemed publided Named, Keemed publided Named, Keemed publided Named, Keemed movelses (Vella-Norca), Named paddd Named, Named movelses (Vella-Norca), Named movelses (Vella-Norca), Named paddd Named, Named movelses (Vella-Norca), Named paddd Named, Named movel Named, Named pell Named, Named movel Named Named Market Named Market Named Name		
prubufe SR, Kreme, Komen prubufe SR, Kremel prubufe SR, Kremi, Komen pumpeting Kreme, Kremi pumpeting Kreme, Kremi pande Komen, Kremel pande Komen, Kremel pande Komen, Kremel pande Komen, Kremel pande Kremi, Kremi pande Kr		
penulusig Komeng, Komenl pumpickidog Komeng, Komenl pumpickidog Komeng, Komenl paddid Komeng, Komeng paddid Komeng, Komeng, Amang, Amang, paddid Komeng, Komeng, Amang, paddid Komeng, Komeng, paddid Komeng, Komeng, paddid Komeng, Komeng, paddid p		
printf SB, Kermi, Kermi novdpa (Erib, Kerci), Kermi novdpa (Erib, Kerci), Kermi novdpa (Erib, Kerci), Kermi novdpa Kermi, Kermi jib 1.7 addi Krid, Kedx capl Krid, Kedx capl Krid, Kedx capl Krid, Kedx capl Krid, Kedx inuli Krild, Kecx inuli Krild, Kecx addi Kerc, (Krba, Krax, 4), Kecx inuli Krild, Kecx addi Kerc, (Krba, Krax, 4) leal 1(Krid), Keax capl Keax, Kridd jib 1.18 movi (Krild, Keax capl Keax, Kridd jib 1.18 movi (Krild, Kridd, Kridd) joe 1.18 movi Kerx, (Krba, Krax, 4) capl Kridd, Kridd, Krba, Krax, 4) 1.18: add 318, Krap cri_member_state cri_def_cri_erfect 48 movq Krhp, Kril norl Keds, Keds popq Krhp cri_def_cri_erfect 22 popq Krid cri_erf_cri_erfect 24 popq Krid cri_erf_cri_erfect 24 popq Krid cri_erf_cri_erfect 15		
pumpckidg Kurme, Kurmel mordaga (Krills, Kricx), Kurmel padded Kurma, Kurmel movaps Kurme, (Krills, Krex) adds \$16, Krex cmpl Kest, Kried jb .17 addl Nred, Kods cmpl Nred, Kods je .18 mov) Kods, Koax mov) (Krills, Krax), Seek inull Nrild, Keek inull Nrild, Keek cmpl Kass, Kried jb .11 mov) (Krills, Krax), Seek addl Morx, (Krbu, Krax, 4), Neck addl Nr. (Krbu, Krax, 4), Nr. (Krbu, Krax, 4) cmpl Nrds, Kradd jbe .118 mov) Nrds, Koax inull (Krll, Krbu, Krax, 4) 1.18: add \$16, Nrbu, Krax, 4) 1.18: add \$16, Nrbu, Krax, 4) 1.18: add \$16, Krbu, Krax, 4) 1.19: add \$16, Krbu, Krax, 4) 1.10: cri_(def_cfa_offset 48 movq Nrbp, Nril xorl Nrds, Kodi popg Nrbu .cfi_def_cfa_offset 48 popg Nrbu .cfi_def_cfa_offset 48 popg Nrbu .cfi_def_cfa_offset 32 popg Nrll .cfi_def_cfa_offset 13 popq Nrll .cfi_def_cfa_offset 16 popq Nrl4 .cfi_def_cfa_offset 18		
morday (Er18, Ercx), Emme padd Exmm1, Kemm8, (Er18, Ercx) adds 516, Ercx cmp1 Kes1, Kr8d jb 1.7 add1 Wr8d, Kedx cmp1 Wr8d, Kedx cmp1 Wr8d, Kedx cmp1 Wr8d, Kedx mov1 Kr12, Krax,4), Mecx mov1 Kr12, Krax,4), Mecx inull Wr1d, Mecx add1 Mecx, (Wrbx, Wrax,4) leal 1(Wrdx), Meax cmp1 Meax, Kr14d jbe 1.18 mov1 (Kr12, Krax,4), Mecx add1 Sp, Medx inull Wr1d, Mecx add1 Sp, Medx inull Kr1d, Mecx add1 Sp, Medx inull (Mr12, Krax,4), Me1d add1 Sp, Medx inull (Mr12, Krax,4), Wr1d add1 Wr1d, (Krbx, Krax,4), Me1d add1 Wr1d, (Kr		
padd# Xem1, Xemme movaps Xemme, (Xr18, Xrcx) adds \$16, Xrcx capl Xesi, Xrdd jb .17 add1 Xrdd, Xedx capl Yrdd, Xedx capl Yrdd, Xedx nov) (Kr12, Xrax, 4), Xecx inull Yr11d, Xecx add1 Xecx, (Krbx, Xrax, 4), Xr11d add1 Xecx add1 Xecx, (Krbx, Xrax, 4), Xr11d add1 Xr11d, (Krbx, Xrax, 4) 1.18: addq \$16, Xrpp .cfi_remember_state .cfi_def_cfa_offset 48 movq Xrbp, Xrn; xorl Xedx, Xedi poop Xrbp .cfi_def_cfa_offset 48 poop Xrbp .cfi_def_cfa_offset 16 poop Xrbq .cfi_def_cfa_offset 16 poop Xrbq .cfi_def_cfa_offset 18		
movaps Yomen, (Evila, News) addq \$16, News cmpl Nesi, Nedd jb .17 addl Nedd, Nedd cmpl Nedd, Nedd je .118 movl Nedd, Nedd ig .118 movl Nedd, Nex addl Necx, (Newb, Nexa, 4) leal 1(Redd), Nex addl Necx, (Newb, Nexa, 4) leal 1(Redd), Nex addl Se, Nedd jbe .118 movl (Nell, Nex, 4), Nex addl Se, Nedd imull Neldd, Necx addl Nex, (Neb, Nexa, 4) cmpl Nedd, Nex addl Nex, (Neb, Nexa, 4) cmpl Nedd, Nex addl Nex, Neld jbe .118 movl Nedd, Nex addl Neld, (Neb, Nexa, 4) ctpl Nedd, Nex addl Neld, (Neb, Nexa, 4) .118: addq \$16, Nesp .cfi_momenbor_state .cfi_def_cfa_offset 48 movq Neby, Neld popq Neb .cfi_def_cfa_offset 48 movq Neby, Neld popq Neld .cfi_def_cfa_offset 48 movq Neby, Neld popq Neld .cfi_def_cfa_offset 22 popq Nell .cfi_def_cfa_offset 24 popq Nell .cfi_def_cfa_offset 24 popq Nell .cfi_def_cfa_offset 26		
addq \$16, %rcz cmpl %rsd, %cdx cmpl %rsd, %cdx cmpl %rsd, %cdx movl %cdx, %cmx movl (%rlz, %rmx, 4), %ccx imull %rlld, %ccx addl %ccx, (%rbx, %rax, 4) lcal 1(%rdx), %cax cmpl %cax, %rsd jbc .18 movl (%rlz, %rmx, 4), %ccx addl \$1, %cdx addl %ccx, (%rbx, %rax, 4) cmpl %cdx, %rldd jbc .18 movl %cdx, %rsd imull %rlld, %ccx addl %ccx, (%rbx, %rax, 4) cmpl %cdx, %rldd jbc .18 movl %cdx, %rax imull %rlld, %rbx, %rax, 4) tl8: addq \$16, %rsp .fi_remember_state .fi_ref_cf_orfset 48 movq %rbp, %rsi xorl %cdi, %edi popq %rbp .fi_ddf_cf_cf_orfset 40 popq %rbp .fi_ddf_cf_orfset 32 popq %rll .fi_ddf_cf_orfset 24 popq %rll .fi_ddf_cf_orfset 24 popq %rll .fi_df_cf_orfset 16 popq %rll .fi_df_cf_of_orfset 16 popq %rll .fi_df_cf_of_offset 16		paddd %xmm1, %xmm0
cmpl Keri, Kedd jb .1.7 addl Mr8d, Kedx cmpl Mr8d, Kedi je .1.8 mov1 (Korl, Meax mov1 (Korl, Meax, 4), Mecx imull Kr3d, Kecx addl Kecx, (Krbx, Krax, 4) leal 1(Krdx), Keax cmpl Meax, Kr14d jbe .1.8 mov1 (Kv12, Krax, 4), Mecx addl Secx frb, Kecx addl Meax, (Krbx, Krax, 4) cmpl Kedx, Kr14d jbe .1.8 mov1 Medx, Kr3d add jie .1.8 mov1 Medx, Keax imull (Kv12, Krax, 4), Kr3d add jie .1.8 add 316, Krap .fi_memeber_state .fi_ef_cf_aoffset 48 movq Mrbp, Krsi xerl Medd, Kedi popt Mrb .fi_def_cfaoffset 40 popt Mrb .fi_def_cfaoffset 40 popt Mrb .fi_def_cfaoffset 32 popt Mrb .fi_def_cf_aoffset 24 popt Mrb .fi_def_cf_aoffset 32 popt Mrb .fi_def_cf_aoffset 36 popt Mrb .fi_def_cf_aoffset 36 popt Mrb .fi_def_cf_aoffset 36		movaps %xmm0, (%r10,%rcx)
jb .17 addl Mr8d, Mcdx cmpl Mr8d, Mcdx cmpl Mr8d, Mcdx cmpl Mr8d, Mcdx cmpl Mr8d, Mcdx movl Mcdx, Mcax movl (Mc12, Mcax, 4), Mccx addl Mccx, (Mrbx, Mrax, 4) leal 1(Mrdx), Mcax cmpl Mcax, Mc14d jbc .110 movl (Mc12, Mrax, 4), Mccx addl \$2, Mcdx addl \$2, Mcdx inull Wr12, Mrax, 4) cmpl Mcdx, Mc14d jbc .110 movl Mcdx, Mcax addl Mccx, (Mrbx, Mrax, 4) cmpl Mcdx, Mc14d jbc .110 movl Mcdx, Mcax inull (Mc12, Mrax, 4), Mr13d addl Mr13d, (Wrbx, Mrax, 4) .110: addq \$16, Mrsp .cfi_cmember_state .cfi_edf_cfa_offsat 48 movg Mrbx, Mc14 movl Mcdx, Mcax incl Mcdx, Mcax movg Mrbx .cfi_def_cfa_offsat 48 popg Mrbx .cfi_def_cfa_offsat 48 popg Mrbx .cfi_def_cfa_offsat 48 popg Mr3d .cfi_def_cfa_offsat 32 popg Mr3d .cfi_def_cfa_offsat 24 popg Mr13 .cfi_def_cfa_offsat 16 popg Mr3d .cfi_def_cfa_offsat 16 popg Mr3d .cfi_def_cfa_offsat 16		addq \$16, %rcx
addl %rdd, %cdx cspl %rdd, %cdx mov1 &cdx, %cax mov1 &cdx, %cax addl %ccx, (%rbx, %rax, 4), %ccx imull %r31d, %ccx addl %ccx, (%rbx, %rax, 4) leal 1(krdx), %eax cspl %cax, %r14d jbc .110 mov1 &cdx, %rax, 4), %ccx addl %ccx, (krbx, %rax, 4) cspl %cdx, %r34d jbc .110 mov1 &cdx, %rax, 4) cspl %cdx, %r34d jbc .110 mov1 %cdx, %rax, 4) cspl %cdx, %rax imull &cdx, &cdx imull &cdx imull &cdx, &cdx imull		cmpl %esi, %r9d
addl %rdd, %cdx cspl %rdd, %cdx mov1 &cdx, %cax mov1 &cdx, %cax addl %ccx, (%rbx, %rax, 4), %ccx imull %r31d, %ccx addl %ccx, (%rbx, %rax, 4) leal 1(krdx), %eax cspl %cax, %r14d jbc .110 mov1 &cdx, %rax, 4), %ccx addl %ccx, (krbx, %rax, 4) cspl %cdx, %r34d jbc .110 mov1 &cdx, %rax, 4) cspl %cdx, %r34d jbc .110 mov1 %cdx, %rax, 4) cspl %cdx, %rax imull &cdx, &cdx imull &cdx imull &cdx, &cdx imull		
cmpl %r8d, %edi je .118 mov1 %cdx, %oax mov1 (%r12,%rax,4), %ecx imull %l3d, %ecx addl %ecx, (%rbx,%rax,4) leal 1(Krdx), %eax cmpl %eax, %r14d jbe .118 mov1 (Kr12,%rax,4), %ecx addl \$2, %edx imull %r13d, %ecx addl %ecx, (%rbx,%rax,4) cmpl %edx, %r14d jbe .118 mov1 %edx, %r14d jbe .118 mov1 %edx, %rax imull (%r12,%rax,4), %r13d addl %r13d, (%rbx,%rax,4), %r13d addl %r13d, (%rbx,%rax,4), %r13d addl %r13d, (%rbx,%rax,4) 1.18: addq \$16, %rap .cfi_remember_state .cfi_def_cfa_off_sets movq %rbp, %rsi xorl %edi, %edi popq %rby .cfi_def_cfa_offset 48 popq %rbp .cfi_def_cfa_offset 42 popq %rbp .cfi_def_cfa_offset 32 popq %r12 .cfi_def_cfa_offset 24 popq %r13 .cfi_def_cfa_offset 24 popq %r13 .cfi_def_cfa_offset 16 popq %r14		-
je .118 movl Xcdx, Kcax movl (Kr12, Krax, 4), Kccx imull Kr13d, Mccx addl Mccx, (Krbx, Krax, 4) leal 1(Krdx), Kcax cmpl Xcax, Kr14d jbc .118 movl (Kr12, Krax, 4), Xccx addl 52, Mcdx imull Kr13d, Mccx addl 52, Mcdx imull Kr13d, Mccx addl Xccx, (Krbx, Krax, 4) cmpl Xcdx, Kr14d jbc .118 movl Mcdx, Kr2d imull (Kr12, Krax, 4), Xr13d addl Tr13d, (Krbx, Krax, 4) tile: add \$16, Krap .cfi, remember, state .cfi_def_cfa_offset 48 movq Xrbp, Krsi xorl Xcdi, Xcdi popq Wrbx .cfi_def_cfa_offset 48 popq Xrbp .cfi_def_cfa_offset 32 popq Xrbp .cfi_def_cfa_offset 32 popq Xrbp .cfi_def_cfa_offset 24 popq Xrla .cfi_def_cfa_offset 24 popq Xrla .cfi_def_cfa_offset 24 popq Xrla .cfi_def_cfa_offset 24 popq Xrla .cfi_def_cfa_offset 16 popq Xrla .cfi_def_cfa_offset 16 popq Xrla		
movl %chz, %axx movl (%chz, %axx, 4), %ccx imull %clad, %ccx, add1 %ccx, (%rbx, %rax, 4) leal 1(%rdx), %cax cmpl %cax, %r14d jbe .110 movl (%clz, %rax, 4), %ccx add1 %ccx, (%rbx, %rax, 4) cmpl %cdx, %r14d jbe .110 movl %cdx, %rbx, %rax, 4) cmpl %cdx, %r14d jbe .110 movl %cdx, %cax imull (%clz, %rax, 4), %r13d add1 %clad, %r13d, (%rbx, %rax, 4) .110: add %clad, %clad imull (%clz, %rax, 4), %r13d add1 %clad, %rbx .cfi_def_cfa_offact 48 movq %rby, %ri xorl %cdi, %cdi popq %rbx .cfi_def_cfa_offact 48 popq %rby .cfi_def_cfa_offact 48		
moul (%-12,%rax,4), %ecx imull %r13d, %ecx addl %ecx, (%rbx,%rax,4) leal 1(%rdx), %eax cmpl %eax, %r14d jbe .110 movl (%r12,%rax,4), %ecx addl 12, %edx imull %r13d, %ecx addl %ecx, (%rbx,%rax,4) cmpl %edx, %rax,4) cmpl %edx, %rax imull (%r12,%rax,4), %r13d addl %r13d, (%rbx,%rax,4) inull %r13d, %ex imull (%r12,%rax,4), %r13d addl %r13d, (%rbx,%rax,4) i.18: addq 516, %rsp .cfi_def_cfa_offset 48 movq %rbp, %rsi xorl %edi, %edi popq %rb .cfi_def_cfa_offset 48 popq %rb .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 16 popq %r14 .cfi_def_cfa_offset 16 popq %r14 .cfi_def_cfa_offset 16		_
imull %r13d, %ccx addl %ccx, (%rbx,%rax,4) leal 1(krdx), %cax cmpl %cax, %r14d jbc .llb movl (%r12,%rax,4), %ccx addl \$2, %cdx imull %r13d, %ccx addl %ccx, (%rbx,%rax,4) cmpl %cdx, %r14d jbc .llb movl %cdx, %rax imull (%r12,%rax,4), %r13d addl %r13d, (%rbx,%rax,4) .llb: addq \$16, %rsp .cfi_remember_state .cfi_def_cfa_offset 48 movg %rby, %rsi xorl %cdi, %cdi popq %rbx .cfi_def_cfa_offset 48 popq %rby .cfi_def_cfa_offset 32 popq %r12 .cfi_def_cfa_offset 24 popq %r13 .cfi_def_cfa_offset 24 popq %r13 .cfi_def_cfa_offset 24 popq %r14 .cfi_def_cfa_offset 16 popq %r14 .cfi_def_cfa_offset 16 popq %r14 .cfi_def_cfa_offset 16		
add1 %ecx, (%rbx,%rax,4) leal 1(%rdx), %cax cmp1 %eax, %rl4d jbe .l18 mov1 (%rl2,%rax,4), %ecx add1 \$2, %edx imull %rl3d, %ecx add1 %ecx, (%rbx,%rax,4) cmp1 %edx, %rl4d jbe .l18 mov1 %edx, %rl4d jbe .l18 mov1 %edx, %rl4d jbe .l18 add1 %rl3d, (%rbx,%rax,4), %rl3d add1 %rl3d, (%rbx,%rax,4), %rl3d add1 %rl3d, (%rbx,%rax,4) .l18: add2 \$16, %rsp .cfi_remember_state .cfi_def_cfa_offset 48 movq %rbp, %rsi xorl %edi, %edi popq %rbx .cfi_def_cfa_offset 40 popq %rby .cfi_def_cfa_offset 32 popq %rl2 .cfi_def_cfa_offset 24 popq %rl3 .cfi_def_cfa_offset 24 popq %rl3 .cfi_def_cfa_offset 16 popq %rl4 .cfi_def_cfa_offset 16 popq %rl4 .cfi_def_cfa_offset 8		
leal 1(%rdx), %eax cmpl %eax, %P14d jbe.1.80 movl (%r12,%rax,4), %ecx add1 \$2, %edx imull %r13d, %ecx add1 %ecx, (%rbx,%rax,4) cmpl %edx, %eax imull (%r12,%rax,4), %r13d add1 %p16.1.80 movl %edx, %eax imull (%r12,%rax,4), %r13d add1 %r13d, %rbx,%rax,4) .L18: addq \$16, %rsp .cfi_remember_state .cfi_def_cfa_offset 48 movq %rbp, %rsi xorl %edi, %edi popq %rbx .cfi_def_cfa_offset 48 popq %rby .cfi_def_cfa_offset 48 popq %rby .cfi_def_cfa_offset 32 popq %r12 .cfi_def_cfa_offset 32 popq %r13 .cfi_def_cfa_offset 24 popq %r13 .cfi_def_cfa_offset 16 popp %r14		
cmpl %cax, %r14d jbc .110 mov1 (%r12,%rax,4), %ccx add1 \$2, %cdx imull %r13d, %ccx add1 %ccx, (%rbx,%rax,4) cmpl %cdx, %c14d jbc .110 mov1 %cdx, %cax imull (%r12,%rax,4), %r13d add1 %r13d, (%rbx,%rax,4) t.100: addq \$16, %rsp .cfi_cmemsber_state .cfi_def_cfa_offset 48 movq %rbp, %rsi xorl %cdi, %cdi popq %rbx .cfi_def_cfa_offset 40 popq %rby .cfi_def_cfa_offset 32 popq %r12 .cfi_def_cfa_offset 24 popq %r13 .cfi_def_cfa_offset 16 popq %r14		
jbe .L10 mov1 (%r1z, %rax, 4), %ecx add1 \$2, %edx imull %r13d, %ecx add2 %ecx, (%rbx, %rax, 4) cmp1 %edx, %r14d jbe .L10 mov1 %edx, %eax imull (%r1z, %rax, 4), %r13d add1 %r13d, (%rbx, %rax, 4) .L10: addq \$16, %rsp .cfi_nemember_state .cfi_def_cfa_offset 48 movq %rbp, %rsi xor1 %edi, %edi popq %rbx .cfi_def_cfa_offset 40 popq %rby .cfi_def_cfa_offset 40 popq %rby .cfi_def_cfa_offset 32 popq %r12 .cfi_def_cfa_offset 24 popq %r13 .cfi_def_cfa_offset 24 popq %r14 .cfi_def_cfa_offset 16 popq %r14 .cfi_def_cfa_offset 16 popq %r14		
movl (%r12,%rax,4), %ecx add1 \$1, %edx imull %r13d, %ecx add1 %ecx, (%rbx,%rax,4) cmpl %edx, %rax, 4) cmpl %edx, %rax, 4) cmpl %edx, %eax imull (%r12,%rax,4), %r13d add1 %r13d, (%rbx,%rax,4) .L10: addq \$16, %rsp .cfi_remember_state .cfi_def_cfa_offset 48 movq %rbp, %rsi xorl %edi, %edi popq %rbx .cfi_def_cfa_offset 48 popq %rby .cfi_def_cfa_offset 48 popq %rby .cfi_def_cfa_offset 48 popq %r12 .cfi_def_cfa_offset 24 popq %r13 .cfi_def_cfa_offset 16 popq %r14 .cfi_def_cfa_offset 16 popq %r14 .cfi_def_cfa_offset 16		cmpl %eax, %r14d
addl \$2, %ecx imull %r13d, %ecx addl %ecx, (%rbx, %rax,4) cmpl %edx, %r14d jbe.L18 movl %edx, %eax imull (%r12, %rax,4), %r13d addl %r13d, (%rbx, %rax,4) .118: addq \$16, %rsp .cfi_remember_state .cfi_dsf_cfa_offset 48 movq %rbp, %rsi xorl %edi, %edi popq %rbx .cfi_def_cfa_offset 40 popq %rby .cfi_def_cfa_offset 22 popq %r12 .cfi_def_cfa_offset 24 popq %r13 .cfi_def_cfa_offset 24 popq %r13 .cfi_def_cfa_offset 16 popq %r14 .cfi_def_cfa_offset 16		jbe .L10
imull %r13d, %ecx add1 %ecx, (%rbx, %rax, 4) cmpl %edx, %r14d jbe .L18 movl %edx, %eax imull (%r12, %rax, 4), %r13d add1 %f13d, (%rbx, %rax, 4) .L18: addq \$16, %rsp .cfi_remember_state .cfi_def_cfa_offset 48 movq %rbp, %rsi xorl %edi, %edi popq %rbx .cfi_def_cfa_offset 48 popq %rbx .cfi_def_cfa_offset 32 popq %r12 .cfi_def_cfa_offset 24 popq %r13 .cfi_def_cfa_offset 24 popq %r14 .cfi_def_cfa_offset 16 popq %r14		movl (%r12,%rax,4), %ecx
addl %ecx, (%rbx,%rax,4) cmpl %edx, %rl4d jbe .L10 movl %edx, %eax imull (%rl2,%rax,4), %rl3d addl %rl3d, (%rbx,%rax,4) .L10: addq \$16, %rsp .cfi_memember_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 %rl2 .cfi_def_cfa_offset 24 popq %rl3 .cfi_def_cfa_offset 16 popq %rl4 .cfi_def_cfa_offset 16		addl \$2, %edx
cmpl %edx, %r14d jbc .L18 movl %edx, %eax imull (%r12,%rax,4), %r13d addl %r13d, (%rbx,%rax,4) L18: addq \$16, %rsp .fi_remember_state .fi_def_cfa_offset 48 movq %rbp, %rsi xorl %edi popq %rbx .fi_def_cfa_offset 40 popq %rbp .fi_def_cfa_offset 32 popq %r12 .fi_def_cfa_offset 24 popq %r13 .fi_def_cfa_offset 24 popq %r13 .fi_def_cfa_offset 16 popq %r14 .fi_def_cfa_offset 16		imull %r13d, %ecx
cmpl %edx, %r14d jbc .L18 movl %edx, %eax imull (%r12,%rax,4), %r13d addl %r13d, (%rbx,%rax,4) L18: addq \$16, %rsp .fi_remember_state .fi_def_cfa_offset 48 movq %rbp, %rsi xorl %edi popq %rbx .fi_def_cfa_offset 40 popq %rbp .fi_def_cfa_offset 32 popq %r12 .fi_def_cfa_offset 24 popq %r13 .fi_def_cfa_offset 24 popq %r13 .fi_def_cfa_offset 16 popq %r14 .fi_def_cfa_offset 16		addl %ecx, (%rbx,%rax,4)
jbe .L18 movl %cdx, %cax imull (%r12,%rax,4), %r13d addl %r13d, (%rbx,%rax,4) .L18: addq \$16, %rsp .cfi_remember_state .cfi_def_cfa_offset 48 movq %rby, %rsi xorl %cdi, %cdi popq %rbx .cfi_def_cfa_offset 48 popq %rbp .cfi_def_cfa_offset 32 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 16		
movl %edx, %eax imull (%n12,%nax,4), %r13d add1 %r13d, (%rbx,%rax,4) .L18: addq \$16, %rsp .cfi_remember_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 .cfi_def_cfa_offset 24 popq %r13 .cfi_def_cfa_offset 16 popq %r14 .cfi_def_cfa_offset 8		
<pre>imull (%r12,%rax,4), %r13d addl %r13d, (%rbx,%rax,4) .Lle: addq \$16, %rsp .cfi_remember_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 .cfi_def_cfa_offset 24 popq %r13 .cfi_def_cfa_offset 16 popq %r14 .cfi_def_cfa_offset 16</pre>		-
addl %r13d, (%rbx,%rax,4) L18: addq \$16, %rsp .fi_remember_state .fi_def_cfa_offset 48 movq %rbp, %rsi xorl %cdi, %edi popq %rbx .fi_def_cfa_offset 48 popq %rbp .cfi_def_cfa_offset 32 popq %rl2 .cfi_def_cfa_offset 24 popq %r13 .cfi_def_cfa_offset 16 popq %r14 .cfi_def_cfa_offset 8		
L18: addq \$16, %rsp .cfi_remember_state .cfi_def_cfa_offset 48 movq %rbp, %rsi xorl %edi, %edi popq %rbx .cfi_def_cfa_offset 48 popq %rbp .cfi_def_cfa_offset 32 popq %r12 .cfi_def_cfa_offset 24 popq %r13 .cfi_def_cfa_offset 16 popq %r14		
addq \$16, %rsp .cfi_remember_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 .cfi_def_cfa_offset 24 popq %r13 .cfi_def_cfa_offset 16 popq %r14 .cfi_def_cfa_offset 8		
.cfi_remember_state .cfi_def_cfa_offset 48 movq %rbp, %rsi xorl %edi, %edi popq %rbx .cfi_def_cfa_offset 48 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		
.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 .cfi_def_cfa_offset 24 popq %r13 .cfi_def_cfa_offset 16 popq %r14 .cfi_def_cfa_offset 8		
movq %rbp, %rsi xorl %edi, %edi popq %rbx .cfi_def_cfa_offset 48 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		
<pre>xorl %edi, %edi popq %rbx .cfi_def_cfa_offset 48 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</pre>		
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		
.cfi_def_cfa_offset 48 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		xorl %edi, %edi
<pre>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</pre>		popq %rbx
<pre>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</pre>		.cfi_def_cfa_offset 40
<pre>.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</pre>		
<pre>popq %r12 .cfi_def_cfa_offset 24 popq %r13 .cfi_def_cfa_offset 16 popq %r14 .cfi_def_cfa_offset 8</pre>		
.cfi_def_cfa_offset 24 popq %r13 .cfi_def_cfa_offset 16 popq %r14 .cfi_def_cfa_offset 8		
<pre>popq %r13 .cfi_def_cfa_offset 16 popq %r14 .cfi_def_cfa_offset 8</pre>		
<pre>.cfi_def_cfa_offset 16 popq %r14 .cfi_def_cfa_offset 8</pre>		
<pre>popq %r14 .cfi_def_cfa_offset 8</pre>		
.cfi_def_cfa_offset 8		
jmp clock_get_ume		
		Jmp clock_Returne
	l.	