codigosBP0.pdf

postdata9

Arquitectura de Computadores

2º Grado en Ingeniería Informática

Escuela Técnica Superior de Ingenierías Informática y de Telecomunicación
Universidad de Granada

KEEP CALM AND ESTUDIA UN POQUITO

WUOLAH + #QuédateEnCasa

Códigos

SumaVectoresLocales.c

```
/* SumaVectoresLocales.c
Suma de dos vectores: v3 = v1 + v2

Para compilar usar (-lrt: real time library):
      gcc  -O2 SumaVectores.c -o SumaVectores --lrt
gcc  -O2 --S SumaVectores.c --lrt   //para generar el código ensamblador

Para ejecutar use: SumaVectoresC longitud
*/

#include <stdlib.h>       // biblioteca con funciones atoi(), malloc() y free()
#include <stdio.h>        // biblioteca donde se encuentra la función printf()
#include <time.h>         // biblioteca donde se encuentra la función clock_gettime()

//#define PRINTF_ALL      // comentar para quitar el printf que imprime todos los componentes
//Sólo puede estar definida una de las tres constantes VECTOR_ (sólo uno de los ...
//tres defines siguientes puede estar descomentado):
#define VECTOR_LOCAL          // descomentar para que los vectores sean variables locales (si se supera el
                              //tamaño de la pila se generará el error "Violación de Segmento")

//#define VECTOR_GLOBAL       // descomentar para que los vectores sean variables globales (su longitud
                              //no estará limitada por el tamaño de la pila del programa)

//#define VECTOR_DYNAMIC      // descomentar para que los vectores sean variables dinámicas (memoria
                              //reutilizable durante la ejecución)
#ifdef VECTOR_GLOBAL
#define MAX 33554432          //=2^25
double v1[MAX], v2[MAX], v3[MAX];
#endif

int main(int argc, char** argv){

        int i;
        struct timespec cgt1,cgt2; double ncgt; //para tiempo de ejecución

        //Leer argumento de entrada (nº de componentes del vector)
        if (argc<2){
                printf("Faltan nº componentes del vector\n");
                exit(-1);
        }

        unsigned int N = atoi(argv[1]); // Máximo N =2^32-1=4294967295 (sizeof(unsigned int) = 4 B)
        #ifdef VECTOR_LOCAL
        double v1[N], v2[N], v3[N];   // Tamaño variable local en tiempo de ejecución disponible en C a
                                      //partir de actualización C99
        #endif

        #ifdef VECTOR_GLOBAL
        if (N>MAX) N=MAX;
        #endif

        #ifdef VECTOR_DYNAMIC
        double *v1, *v2, *v3;
        v1 = (double*) malloc(N*sizeof(double));  // malloc necesita el tamaño en bytes
        v2 = (double*) malloc(N*sizeof(double));  //si no hay espacio suficiente malloc devuelve NULL
```

```
	v3 = (double*) malloc(N*sizeof(double));
	if ( (v1==NULL) || (v2==NULL) || (v3==NULL) ){
		printf("Error en la reserva de espacio para los vectores\n");
		exit(-2);
	}
	#endif

	//Inicializar vectores
	for(i = 0; i < N; i++){
		v1[i] = N*0.1+i*0.1; v2[i] = N*0.1-i*0.1; //los valores dependen de N
	}

	clock_gettime(CLOCK_REALTIME,&cgt1);
	//Calcular suma de vectores
	for(i=0; i<N; i++)
		v3[i] = v1[i] + v2[i];

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

	//Imprimir resultado de la suma y el tiempo de ejecución
	#ifdef PRINTF_ALL
	printf("Tiempo(seg.):%11.9f\t / Tamaño Vectores:%u\n",ncgt,N);
	for(i=0; i<N; i++)
		printf("/ V1[%d]+V2[%d]=V3[%d](%8.6f+%8.6f=%8.6f) /\n", i,i,i,v1[i],v2[i],v3[i]);

	#else
	printf("Tiempo(seg.):%11.9f\t / Tamaño Vectores:%u\t/ V1[0]+V2[0]=V3[0](%8.6f+%8.6f
	%8.6f) / / V1[%d]+V2[%d]=V3[%d](%8.6f+%8.6f=%8.6f) /\n", ncgt,N,v1[0],v2[0],v3[0],N-1,N-
	1,N-1,v1[N-1],v2[N-1],v3[N-1]);
	#endif

	#ifdef VECTOR_DYNAMIC
	free(v1); // libera el espacio reservado para v1
	free(v2); // libera el espacio reservado para v2
	free(v3); // libera el espacio reservado para v3
	#endif

	return 0;
}
```

SumaVectoresLocales.sh

```
#!/bin/bash
#Se asigna al trabajo el nombre SumaVectores_vlocales
#PBS -N SumaVectoresC_vlocales
#Se asigna al trabajo la cola ac
#PBS -q ac
#Se imprime información del trabajo usando variables de entorno de PBS

#echo "Id. usuario del trabajo: $PBS_O_LOGNAME"
#echo "Id. del trabajo: $PBS_JOBID"
#echo "Nombre del trabajo especificado por usuario: $PBS_JOBNAME"
#echo "Nodo que ejecuta qsub: $PBS_O_HOST"
#echo "Directorio en el que se ha ejecutado qsub: $PBS_O_WORKDIR"
#echo "Cola: $PBS_QUEUE"
#echo "Nodos asignados al trabajo:"
#cat $PBS_NODEFILE

#Se ejecuta SumaVectorC, que está en el directorio en el que se ha ejecutado qsub,
#para N potencia de 2 desde 2^16 a 2^26
for((N = 65536; N < 67108865; N=N*2))
do
        ./SumaVectoresLoc $N
done
```

SumaVectores_vlocales.o

```
Tiempo(seg.):0.000420133        / Tamaño Vectores:65536        / V1[0]+V2[0]=V3[0]
(6553.600000+6553.600000=13107.200000) / / V1[65535]+V2[65535]=V3[65535]
(13107.100000+0.100000=13107.200000) /

Tiempo(seg.):0.000839202        / Tamaño Vectores:131072       / V1[0]+V2[0]=V3[0]
(13107.200000+13107.200000=26214.400000) / / V1[131071]+V2[131071]=V3[131071]
(26214.300000+0.100000=26214.400000) /

Tiempo(seg.):0.001392156        / Tamaño Vectores:262144       / V1[0]+V2[0]=V3[0]
(26214.400000+26214.400000=52428.800000) / / V1[262143]+V2[262143]=V3[262143]
(52428.700000+0.100000=52428.800000) /
```

SumaVectores_vlocales.e

```
/var/lib/torque/mom_priv/jobs/63627.atcgrid.SC: line 23: 29811 Segmentation fault      (core dumped) ./
SumaVectoresLoc $N

/var/lib/torque/mom_priv/jobs/63627.atcgrid.SC: line 23: 29814 Segmentation fault      (core dumped) ./
SumaVectoresLoc $N

/var/lib/torque/mom_priv/jobs/63627.atcgrid.SC: line 23: 29817 Segmentation fault      (core dumped) ./
SumaVectoresLoc $N

/var/lib/torque/mom_priv/jobs/63627.atcgrid.SC: line 23: 29824 Segmentation fault      (core dumped) ./
SumaVectoresLoc $N

/var/lib/torque/mom_priv/jobs/63627.atcgrid.SC: line 23: 29827 Segmentation fault      (core dumped) ./
SumaVectoresLoc $N

/var/lib/torque/mom_priv/jobs/63627.atcgrid.SC: line 23: 29830 Segmentation fault      (core dumped) ./
SumaVectoresLoc $N

/var/lib/torque/mom_priv/jobs/63627.atcgrid.SC: line 23: 29834 Segmentation fault      (core dumped) ./
SumaVectoresLoc $N

/var/lib/torque/mom_priv/jobs/63627.atcgrid.SC: line 23: 29837 Segmentation fault      (core dumped) ./
SumaVectoresLoc $N
```

Vectores Globales

SumaVectoresGlobales.c

```
/* SumaVectoresLocales.c
Suma de dos vectores: v3 = v1 + v2

Para compilar usar (-lrt: real time library):
      gcc  -O2 SumaVectores.c -o SumaVectores --lrt
gcc  -O2 --S SumaVectores.c --lrt   //para generar el código ensamblador

Para ejecutar use: SumaVectoresC longitud
*/

#include <stdlib.h>       // biblioteca con funciones atoi(), malloc() y free()
#include <stdio.h>        // biblioteca donde se encuentra la función printf()
#include <time.h>         // biblioteca donde se encuentra la función clock_gettime()

//#define PRINTF_ALL      // comentar para quitar el printf que imprime todos los componentes
//Sólo puede estar definida una de las tres constantes VECTOR_ (sólo uno de los ...
//tres defines siguientes puede estar descomentado):
//#define VECTOR_LOCAL      // descomentar para que los vectores sean variables locales (si se supera el
                            //tamaño de la pila se generará el error "Violación de Segmento")

#define VECTOR_GLOBAL       // descomentar para que los vectores sean variables globales (su longitud
                            //no estará limitada por el tamaño de la pila del programa)

//#define VECTOR_DYNAMIC  // descomentar para que los vectores sean variables dinámicas (memoria
                            //reutilizable durante la ejecución)
#ifdef VECTOR_GLOBAL
#define MAX 424967295       //=2^32
double v1[MAX], v2[MAX], v3[MAX];
#endif

int main(int argc, char** argv){

        int i;
        struct timespec cgt1,cgt2; double ncgt; //para tiempo de ejecución

        //Leer argumento de entrada (nº de componentes del vector)
        if (argc<2){
                printf("Faltan nº componentes del vector\n");
                exit(-1);
        }

        unsigned int N = atoi(argv[1]); // Máximo N =2^32-1=4294967295 (sizeof(unsigned int) = 4 B)
        #ifdef VECTOR_LOCAL
        double v1[N], v2[N], v3[N];   // Tamaño variable local en tiempo de ejecución disponible en C a
                                      //partir de actualización C99
        #endif

        #ifdef VECTOR_GLOBAL
        if (N>MAX) N=MAX;
        #endif

        #ifdef VECTOR_DYNAMIC
        double *v1, *v2, *v3;
        v1 = (double*) malloc(N*sizeof(double));  // malloc necesita el tamaño en bytes
        v2 = (double*) malloc(N*sizeof(double));  //si no hay espacio suficiente malloc devuelve NULL
```

```
	v3 = (double*) malloc(N*sizeof(double));
	if ( (v1==NULL) || (v2==NULL) || (v3==NULL) ){
		printf("Error en la reserva de espacio para los vectores\n");
		exit(-2);
	}
	#endif

	//Inicializar vectores
	for(i = 0; i < N; i++){
		v1[i] = N*0.1+i*0.1; v2[i] = N*0.1-i*0.1; //los valores dependen de N
	}

	clock_gettime(CLOCK_REALTIME,&cgt1);
	//Calcular suma de vectores
	for(i=0; i<N; i++)
		v3[i] = v1[i] + v2[i];

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

	//Imprimir resultado de la suma y el tiempo de ejecución
	#ifdef PRINTF_ALL
	printf("Tiempo(seg.):%11.9f\t / Tamaño Vectores:%u\n",ncgt,N);
	for(i=0; i<N; i++)
		printf("/ V1[%d]+V2[%d]=V3[%d](%8.6f+%8.6f=%8.6f) /\n", i,i,i,v1[i],v2[i],v3[i]);

	#else
	printf("Tiempo(seg.):%11.9f\t / Tamaño Vectores:%u\t/ V1[0]+V2[0]=V3[0](%8.6f+%8.6f
	%8.6f) / / V1[%d]+V2[%d]=V3[%d](%8.6f+%8.6f=%8.6f) /\n", ncgt,N,v1[0],v2[0],v3[0],N-1,N-
	1,N-1,v1[N-1],v2[N-1],v3[N-1]);
	#endif

	#ifdef VECTOR_DYNAMIC
	free(v1); // libera el espacio reservado para v1
	free(v2); // libera el espacio reservado para v2
	free(v3); // libera el espacio reservado para v3
	#endif

	return 0;
}
```

SumaVectoresGlobales.sh

```
#!/bin/bash
#Se asigna al trabajo el nombre SumaVectores_vglobales
#PBS -N SumaVectoresC_vglobales
#Se asigna al trabajo la cola ac
#PBS -q ac
#Se imprime información del trabajo usando variables de entorno de PBS

#echo "Id. usuario del trabajo: $PBS_O_LOGNAME"
#echo "Id. del trabajo: $PBS_JOBID"
#echo "Nombre del trabajo especificado por usuario: $PBS_JOBNAME"
#echo "Nodo que ejecuta qsub: $PBS_O_HOST"
#echo "Directorio en el que se ha ejecutado qsub: $PBS_O_WORKDIR"
#echo "Cola: $PBS_QUEUE"
#echo "Nodos asignados al trabajo:"
#cat $PBS_NODEFILE

#Se ejecuta SumaVectorC, que está en el directorio en el que se ha ejecutado qsub,
#para N potencia de 2 desde 2^16 a 2^26
for((N = 65536; N < 4294967295; N=N*2))
do
        ./SumaVectoresGlob $N
done
```

SumaVectores_vglobales.o

```
Tiempo(seg.):0.000510229        / Tamaño Vectores:65536        / V1[0]+V2[0]=V3[0]
(6553.600000+6553.600000=13107.200000) / / V1[65535]+V2[65535]=V3[65535]
(13107.100000+0.100000=13107.200000) /

Tiempo(seg.):0.000792256        / Tamaño Vectores:131072       / V1[0]+V2[0]=V3[0]
(13107.200000+13107.200000=26214.400000) / / V1[131071]+V2[131071]=V3[131071]
(26214.300000+0.100000=26214.400000) /

Tiempo(seg.):0.001385945        / Tamaño Vectores:262144       / V1[0]+V2[0]=V3[0]
(26214.400000+26214.400000=52428.800000) / / V1[262143]+V2[262143]=V3[262143]
(52428.700000+0.100000=52428.800000) /

Tiempo(seg.):0.003096655        / Tamaño Vectores:524288       / V1[0]+V2[0]=V3[0]
(52428.800000+52428.800000=104857.600000) / / V1[524287]+V2[524287]=V3[524287]
(104857.500000+0.100000=104857.600000) /

Tiempo(seg.):0.006250664        / Tamaño Vectores:1048576      / V1[0]+V2[0]=V3[0]
(104857.600000+104857.600000=209715.200000) / / V1[1048575]+V2[1048575]=V3[1048575]
(209715.100000+0.100000=209715.200000) /

Tiempo(seg.):0.012137120        / Tamaño Vectores:2097152      / V1[0]+V2[0]=V3[0]
(209715.200000+209715.200000=419430.400000) / / V1[2097151]+V2[2097151]=V3[2097151]
(419430.300000+0.100000=419430.400000) /

Tiempo(seg.):0.025346216        / Tamaño Vectores:4194304      / V1[0]+V2[0]=V3[0]
(419430.400000+419430.400000=838860.800000) / / V1[4194303]+V2[4194303]=V3[4194303]
(838860.700000+0.100000=838860.800000) /

Tiempo(seg.):0.047970985        / Tamaño Vectores:8388608      / V1[0]+V2[0]=V3[0]
(838860.800000+838860.800000=1677721.600000) / / V1[8388607]+V2[8388607]=V3[8388607]
(1677721.500000+0.100000=1677721.600000) /

Tiempo(seg.):0.095998309        / Tamaño Vectores:16777216   / V1[0]+V2[0]=V3[0]
(1677721.600000+1677721.600000=3355443.200000) / /
V1[16777215]+V2[16777215]=V3[16777215](3355443.100000+0.100000=3355443.200000) /
```

```
Tiempo(seg.):0.188855907        / Tamaño Vectores:33554432  / V1[0]+V2[0]=V3[0]
(3355443.200000+3355443.200000=6710886.400000) / /
V1[33554431]+V2[33554431]=V3[33554431](6710886.300000+0.100000=6710886.400000) /

Tiempo(seg.):0.372082705        / Tamaño Vectores:67108864  / V1[0]+V2[0]=V3[0]
(6710886.400000+6710886.400000=13421772.800000) / /
V1[67108863]+V2[67108863]=V3[67108863](13421772.700000+0.100000=13421772.800000) /
```

SumaVectores_vglobales.e

```
(vacío)
```

SumaVectoresDinamicos.c

```
/* SumaVectoresDinamicos.c
Suma de dos vectores: v3 = v1 + v2

Para compilar usar (-lrt: real time library):
      gcc  -O2 SumaVectores.c -o SumaVectores --lrt
gcc  -O2 --S SumaVectores.c --lrt   //para generar el código ensamblador

Para ejecutar use: SumaVectoresC longitud
*/

#include <stdlib.h>       // biblioteca con funciones atoi(), malloc() y free()
#include <stdio.h>        // biblioteca donde se encuentra la función printf()
#include <time.h>         // biblioteca donde se encuentra la función clock_gettime()

//#define PRINTF_ALL      // comentar para quitar el printf que imprime todos los componentes
//Sólo puede estar definida una de las tres constantes VECTOR_ (sólo uno de los ...
//tres defines siguientes puede estar descomentado):
//#define VECTOR_LOCAL      // descomentar para que los vectores sean variables locales (si se supera el
                            //tamaño de la pila se generará el error "Violación de Segmento")

//#define VECTOR_GLOBAL     // descomentar para que los vectores sean variables globales (su longitud
                            //no estará limitada por el tamaño de la pila del programa)

#define VECTOR_DYNAMIC      // descomentar para que los vectores sean variables dinámicas (memoria
                            //reutilizable durante la ejecución)
#ifdef VECTOR_GLOBAL
#define MAX 67108865        //=2^26
double v1[MAX], v2[MAX], v3[MAX];
#endif

int main(int argc, char** argv){

        int i;
        struct timespec cgt1,cgt2; double ncgt; //para tiempo de ejecución

        //Leer argumento de entrada (nº de componentes del vector)
        if (argc<2){
                printf("Faltan nº componentes del vector\n");
                exit(-1);
        }

        unsigned int N = atoi(argv[1]); // Máximo N =2^32-1=4294967295 (sizeof(unsigned int) = 4 B)
        #ifdef VECTOR_LOCAL
        double v1[N], v2[N], v3[N];   // Tamaño variable local en tiempo de ejecución disponible en C a
                                      //partir de actualización C99
        #endif

        #ifdef VECTOR_GLOBAL
        if (N>MAX) N=MAX;
        #endif

        #ifdef VECTOR_DYNAMIC
        double *v1, *v2, *v3;
        v1 = (double*) malloc(N*sizeof(double));  // malloc necesita el tamaño en bytes
        v2 = (double*) malloc(N*sizeof(double));  //si no hay espacio suficiente malloc devuelve NULL
```

```
	v3 = (double*) malloc(N*sizeof(double));
	if ( (v1==NULL) || (v2==NULL) || (v3==NULL) ){
		printf("Error en la reserva de espacio para los vectores\n");
		exit(-2);
	}
	#endif

	//Inicializar vectores
	for(i = 0; i < N; i++){
		v1[i] = N*0.1+i*0.1; v2[i] = N*0.1-i*0.1; //los valores dependen de N
	}

	clock_gettime(CLOCK_REALTIME,&cgt1);
	//Calcular suma de vectores
	for(i=0; i<N; i++)
		v3[i] = v1[i] + v2[i];

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

	//Imprimir resultado de la suma y el tiempo de ejecución
	#ifdef PRINTF_ALL
	printf("Tiempo(seg.):%11.9f\t / Tamaño Vectores:%u\n",ncgt,N);
	for(i=0; i<N; i++)
		printf("/ V1[%d]+V2[%d]=V3[%d](%8.6f+%8.6f=%8.6f) /\n", i,i,i,v1[i],v2[i],v3[i]);

	#else
	printf("Tiempo(seg.):%11.9f\t / Tamaño Vectores:%u\t/ V1[0]+V2[0]=V3[0](%8.6f+%8.6f
	%8.6f) / / V1[%d]+V2[%d]=V3[%d](%8.6f+%8.6f=%8.6f) /\n", ncgt,N,v1[0],v2[0],v3[0],N-1,N-
	1,N-1,v1[N-1],v2[N-1],v3[N-1]);
	#endif

	#ifdef VECTOR_DYNAMIC
	free(v1); // libera el espacio reservado para v1
	free(v2); // libera el espacio reservado para v2
	free(v3); // libera el espacio reservado para v3
	#endif

	return 0;
}
```

SumaVectoresDinamicos.sh

```
#!/bin/bash
#Se asigna al trabajo el nombre SumaVectores_vdinamicos
#PBS -N SumaVectoresC_vdinamicos
#Se asigna al trabajo la cola ac
#PBS -q ac
#Se imprime información del trabajo usando variables de entorno de PBS

#echo "Id. usuario del trabajo: $PBS_O_LOGNAME"
#echo "Id. del trabajo: $PBS_JOBID"
#echo "Nombre del trabajo especificado por usuario: $PBS_JOBNAME"
#echo "Nodo que ejecuta qsub: $PBS_O_HOST"
#echo "Directorio en el que se ha ejecutado qsub: $PBS_O_WORKDIR"
#echo "Cola: $PBS_QUEUE"
#echo "Nodos asignados al trabajo:"
#cat $PBS_NODEFILE

#Se ejecuta SumaVectorC, que está en el directorio en el que se ha ejecutado qsub,
#para N potencia de 2 desde 2^16 a 2^26
for((N = 65536; N < 67108865; N=N*2))
do
        ./SumaVectoresDin $N
done
```

SumaVectores_vlocales.o

```
Tiempo(seg.):0.000415983        / Tamaño Vectores:65536        / V1[0]+V2[0]=V3[0]
(6553.600000+6553.600000=13107.200000) / / V1[65535]+V2[65535]=V3[65535]
(13107.100000+0.100000=13107.200000) /

Tiempo(seg.):0.000841472        / Tamaño Vectores:131072       / V1[0]+V2[0]=V3[0]
(13107.200000+13107.200000=26214.400000) / / V1[131071]+V2[131071]=V3[131071]
(26214.300000+0.100000=26214.400000) /

Tiempo(seg.):0.001685759        / Tamaño Vectores:262144       / V1[0]+V2[0]=V3[0]
(26214.400000+26214.400000=52428.800000) / / V1[262143]+V2[262143]=V3[262143]
(52428.700000+0.100000=52428.800000) /

Tiempo(seg.):0.002581680        / Tamaño Vectores:524288       / V1[0]+V2[0]=V3[0]
(52428.800000+52428.800000=104857.600000) / / V1[524287]+V2[524287]=V3[524287]
(104857.500000+0.100000=104857.600000) /

Tiempo(seg.):0.005996004        / Tamaño Vectores:1048576      / V1[0]+V2[0]=V3[0]
(104857.600000+104857.600000=209715.200000) / / V1[1048575]+V2[1048575]=V3[1048575]
(209715.100000+0.100000=209715.200000) /

Tiempo(seg.):0.012017980        / Tamaño Vectores:2097152      / V1[0]+V2[0]=V3[0]
(209715.200000+209715.200000=419430.400000) / / V1[2097151]+V2[2097151]=V3[2097151]
(419430.300000+0.100000=419430.400000) /

Tiempo(seg.):0.023357001        / Tamaño Vectores:4194304      / V1[0]+V2[0]=V3[0]
(419430.400000+419430.400000=838860.800000) / / V1[4194303]+V2[4194303]=V3[4194303]
(838860.700000+0.100000=838860.800000) /

Tiempo(seg.):0.046919809        / Tamaño Vectores:8388608      / V1[0]+V2[0]=V3[0]
(838860.800000+838860.800000=1677721.600000) / / V1[8388607]+V2[8388607]=V3[8388607]
(1677721.500000+0.100000=1677721.600000) /

Tiempo(seg.):0.094043710        / Tamaño Vectores:16777216   / V1[0]+V2[0]=V3[0]
(1677721.600000+1677721.600000=3355443.200000) / /
V1[16777215]+V2[16777215]=V3[16777215](3355443.100000+0.100000=3355443.200000) /
```

Tiempo(seg.):0.190939352 / Tamaño Vectores:33554432 / V1[0]+V2[0]=V3[0](3355443.200000+3355443.200000=6710886.400000) / / V1[33554431]+V2[33554431]=V3[33554431](6710886.300000+0.100000=6710886.400000) /

Tiempo(seg.):0.371185706 / Tamaño Vectores:67108864 / V1[0]+V2[0]=V3[0](6710886.400000+6710886.400000=13421772.800000) / / V1[67108863]+V2[67108863]=V3[67108863](13421772.700000+0.100000=13421772.800000) /

SumaVectores_vlocales.e

(vacío)