Ejercicio 1, PopCount:

```c
/*
 ============================================================================
 Name        : Peso_popcount_C.c
 Author      : Alex
 Version     :
 Copyright   : Your copyright notice
 Description : Hello World in C, Ansi-style
 ============================================================================
 */

//gcc -m32 -O1 -fno-omit-frame-pointer pesopopcount_C.c -o pesopopcount_C
//gcc -O0 -g -Wall -m32 -fno-omit-frame-pointer
#define TEST 0
#define COPY_PASTE_CALC 0

#if ! TEST
        #define NBITS 20
        #define SIZE (1<<NBITS) //Tamaño suficiente para tiempo apreciable
        unsigned lista[SIZE];
        #define RESULT 10485760
#else
        #define SIZE 4
        unsigned lista[SIZE] = {0x80000000, 0x00100000, 0x00000800, 0x00000001};
        #define RESULT 4
#endif

#include <stdio.h>      // para printf()
#include <stdlib.h>     // para exit()
#include <sys/time.h>   // para gettimeofday(), struct timeval
#define WSIZE 8*sizeof(int)
//#define SIZE (1<<20)  // tamaño suficiente para tiempo apreciable
//unsigned lista[SIZE]; // = { 0x01010101 }; // 0x00000003, 0x00000003};
int resultado = 0;

int popcount1(unsigned* array, int len) {
        int i, k;
        int result = 0;
        for (k = 0; k < len; k++)
                for (i = 0; i < WSIZE; i++) {
                        unsigned mask = 1 << i;
                        result += (array[k] & mask) != 0;
                }
        return result;
}

int popcount2(unsigned* array, int len) {
        int result = 0;
        int i;
        unsigned x;
        for (i = 0; i < len; i++) {
                x = array[i];
                while (x) {
                        result += x & 0x1;
                        x >>= 1;
                }
```

```
        }
        return result;
}

int popcount3(unsigned* array, int len) {
        int result = 0;
        int i;
        unsigned x;
        for (i = 0; i < len; i++) {
                x = array[i];
                asm("\n"
                                "ini3:                                  \n\t"
                                "shr $0x1, %[x]         \n\t" //Desplazar afecta a
CF ZF
                                "adc $0x0, %[r]         \n\t"
                                "test %[x], %[x]        \n\t"
                                "jnz ini3 "

                                : [r] "+r" (result)// e/s: inicialmente 0, salida
valor final
                                : [x] "r" (x) );
        }
        return result;
}

/*
 * Versión C de CS:APP
 */
int popcount4(unsigned* array, int len) {

        int i, k;
        int result = 0;
        for (i = 0; i < len; i++) {
                int val = 0;
                unsigned x = array[i];
                for (k = 0; k < 8; k++) {
                        val += x & 0x01010101; //00000001 00000001 00000001
00000001
                        x >>= 1;
                }
                //val += (val >> 32);
                val += (val >> 16);
                val += (val >> 8);
                result += (val & 0xff);
        }
        return result;
}

/**
 * Versión SSSE3 (pshufb) web http:/wm.ite.pl/articles/sse-popcount.html
 */
int popcount5(unsigned* array, int len) {
        int i;
        int val, result = 0;
        int SSE_mask[] = { 0x0f0f0f0f, 0x0f0f0f0f, 0x0f0f0f0f, 0x0f0f0f0f };
        int SSE_LUTb[] = { 0x02010100, 0x03020201, 0x03020201, 0x04030302 };

        if (len & 0x3)
```

```
                printf("leyendo 128b pero len no múltiplo de 4?\n");
        for (i = 0; i < len; i += 4) {
                asm("movdqu        %[x], %%xmm0 \n\t"
                                "movdqa  %%xmm0, %%xmm1 \n\t" // dos copias de x
                                "movdqu    %[m], %%xmm6 \n\t"// máscara
                                "psrlw           $4, %%xmm1 \n\t"
                                "pand    %%xmm6, %%xmm0 \n\t"// xmm0 – nibbles
inferiores
                                "pand    %%xmm6, %%xmm1 \n\t"// xmm1 – nibbles
superiores

                                "movdqu    %[l], %%xmm2 \n\t"// ...como pshufb
sobrescribe LUT
                                "movdqa  %%xmm2, %%xmm3 \n\t"// ...queremos 2
copias
                                "pshufb  %%xmm0, %%xmm2 \n\t"// xmm2 = vector
popcount inferiores
                                "pshufb  %%xmm1, %%xmm3 \n\t"// xmm3 = vector
popcount superiores

                                "paddb   %%xmm2, %%xmm3 \n\t"// xmm3 - vector
popcount bytes
                                "pxor    %%xmm0, %%xmm0 \n\t"// xmm0 = 0,0,0,0
                                "psadbw  %%xmm0, %%xmm3 \n\t"//xmm3 = [pcnt
bytes0..7|pcnt bytes8..15]
                                "movhlps %%xmm3, %%xmm0 \n\t"//xmm3 = [
0           |pcnt bytes0..7 ]
                                "paddd   %%xmm3, %%xmm0 \n\t"//xmm0 = [ no usado
|pcnt bytes0..15]
                                "movd    %%xmm0, %[val] \n\t"
                                : [val]"=r" (val)
                                : [x] "m" (array[i]),
                                [m] "m" (SSE_mask[0]),
                                [l] "m" (SSE_LUTb[0])
                );
                result += val;
        }
        return result;
}
/**
 * Versión SSE4.2 (popcount)
 */
int popcount6(unsigned* array, int len) {
        int i;
        unsigned x;
        int val, result = 0;
        for (i = 0; i < len; i++) {
                x = array[i];
                asm("popcnt %[x], %[val]        \n\t"
                                : [val]"=r"(val)
                                : [x] "r" (x)
                );
                result += val;
        }
        return result;
}

int popcount7(unsigned* array, int len) {
```

```
        int i;
        unsigned x1, x2;
        int val, result = 0;
        if (len & 0x1)
                printf("Leer 64b y len impar?\n");
        for (i = 0; i < len; i += 2) {
                x1 = array[i];
                x2 = array[i + 1];
                asm("popcnt %[x1], %[val]       \n\t"
                                "popcnt %[x2], %%edi    \n\t"
                                "add    %%edi, %[val]   \n\t"
                                : [val]"=r"(val)
                                : [x1] "r" (x1),
                                [x2] "r" (x2)
                                : "edi"
                );
                result += val;
        }
        return result;
}

void crono(int (*func)(), char* msg) {
        struct timeval tv1, tv2; // gettimeofday() secs-usecs
        long tv_usecs; // y sus cuentas

        gettimeofday(&tv1, NULL);
        resultado = func(lista, SIZE);
        gettimeofday(&tv2, NULL);

        tv_usecs = (tv2.tv_sec - tv1.tv_sec) * 1E6 + (tv2.tv_usec - tv1.tv_usec);
#if ! COPY_PASTE_CALC
        printf("resultado = %d\t", resultado);
        printf("%s:%9ld us\n", msg, tv_usecs);
#else
        printf("%9ld us\n", tv_usecs);
#endif
}

int main() {
#if ! TEST
        int i; // inicializar array
        for (i = 0; i < SIZE; i++) // se queda en cache
                lista[i] = i;
#endif
        crono(popcount1, "popcount1 (    en lenguaje C for  )");
        crono(popcount2, "popcount2 (    en lenguaje C whi  )");
        crono(popcount3, "popcount3 (    Ahorrando máscara  )");
        crono(popcount4, "popcount4 (Sumando bytes completos)");
        crono(popcount5, "popcount5 (        SSSE3          )");
        crono(popcount6, "popcount6 (        SSSE4.2        )");
        crono(popcount7, "popcount7 (        SSSE4.2 64b    )");
//#if ! COPY_PASTE_CALC
//        printf("calculado = %d\n", RESULT);
//#endif
//printf("N*(N+1)/2 = %d\n", (SIZE-1)*(SIZE/2)); /*OF*/
//printf("N*(N+1)/2 = %d\n", sizeof(long));
        exit(0);
}
```

Ejercicio 2 PARITY

```
/*
 ============================================================================
 Name        : Paridad.c
 Author      : Alejandro
 Version     :
 Copyright   :
 Description : Hello World in C, Ansi-style
 ============================================================================
 */

/*gcc -m32 -O1 -fno-omit-frame-pointer pesopopcount_C.c -o pesopopcount_C*/
/*gcc -O0 -g -Wall -Wextra -Wpedantic -m32 -fno-omit-frame-pointer*/
#define TEST 0
#define COPY_PASTE_CALC 0

#if ! TEST
#define NBITS 20
#define SIZE (1<<NBITS) /*Tamaño suficiente para tiempo apreciable*/
unsigned lista[SIZE];
#define RESULT 10485760
#else
#define SIZE 8
//unsigned lista[SIZE] = {0x80000000, 0x00100000, 0x00000800, 0x00000001};
//unsigned lista[SIZE] = {0x7fffffff, 0xffefffff, 0xfffff7ff, 0xfffffffe,
0x01000024,
//    0x00356700, 0x8900ac00, 0x00bd00ef};
#endif

#include <stdio.h>      // para printf()
#include <stdlib.h>     // para exit()
#include <sys/time.h>   // para gettimeofday(), struct timeval
#define WSIZE 8*sizeof(int)
//#define SIZE (1<<20)  // tamaño suficiente para tiempo apreciable
//unsigned lista[SIZE]; // = { 0x01010101 }; // 0x00000003, 0x00000003};
int resultado = 0;
```

```c
int paridad1(unsigned* array, int len) {
	unsigned i, j;
	int paridad;
	unsigned entero;
	int result = 0;

	for (i = 0; i < len; i++) {
		paridad = 0;
		entero = array[i];
		for (j = 0; j < WSIZE; j++) {
			paridad ^= (entero & 1);
			entero >>= 1;
		}
		result += paridad & 0x01;
	}
	return result;
}

int paridad2(unsigned* array, int len) {
	int i;
	int paridad;
	unsigned entero;
	int result = 0;

	for (i = 0; i < len; i++) {
		paridad = 0;
		entero = array[i];
		while (entero) {
			paridad ^= (entero & 1);
			entero >>= 1;
		}
		result += paridad & 0x1;
	}
	return result;
}

/*
 * Versión C de CS:APP Ejercicio 3.22
```

```
 */
int paridad3(unsigned* array, int len) {

      int val = 0;
      int i;
      unsigned x;
      int result = 0;
      for (i = 0; i < len; i++) {
            x = array[i];
            while (x) {
                  val ^= x;
                  x >>= 1;
            }
            result += val & 0x1;
      }
      return result;

}

/*
 * Versión C de CS:APP Ejercicio 3.22 (Traduciendo while)
 */
int paridad4(unsigned* array, int len) {

      int val;
      int i;
      unsigned x;
      int result = 0;

      for (i = 0; i < len; i++) {
            x = array[i];
            val = 0;
            asm(
                        "ini3:                                    \n\t"
                        "xor %[x], %[v]               \n\t"
                        "shr $1, %[x]                 \n\t"
                        "test %[x], %[x]        \n\t"
                        "jnz ini3                     \n\t"
```

```
                        : [v]"+r"(val) // e/s: inicialemnte 0, salida valor final
                        : [x]"r"(x)// entrada: valor del elemento
            );
            result += val & 0x1;
      }
      return result;

}

/**
 * Sumando en árbol
 */
int paridad5(unsigned* array, int len) {

      int i, k;
      int result = 0;
      unsigned x;
      for (i = 0; i < len; i++) {
            x = array[i];
            for (k = 16; k == 1; k /= 2)
                  x ^= x >> k;
            result += (x & 0x01);
      }
      return result;

}

int paridad6(unsigned* array, int len) {
      int j;
      unsigned entero = 0;


      int resultado = 0;

      for (j = 0; j < len; j++) { //Cuando acabe de recorrer el vector se saldrá
del bucle

            entero = array[j]; //Cargo en entero el siguiente numero de la lista
            asm(
```

```
                        "mov  %[x],       %%edx       \n\t"
                        "shr  $16,  %%edx       \n\t"
                        "xor  %[x], %%edx       \n\t"
                        "xor  %%dh, %%dl        \n\t"
                        "setpo  %%dl                        \n\t"
                        "movzx      %%dl, %[x]        \n\t"
                        : [x] "+r" (entero) // input
                        :
                        : "edx"//Clobber
            );
            resultado += entero;
      }
      return resultado;
}

void crono(int (*func)(), char* msg) {
      struct timeval tv1, tv2; // gettimeofday() secs-usecs
      long tv_usecs; // y sus cuentas

      gettimeofday(&tv1, NULL);
      resultado = func(lista, SIZE);
      gettimeofday(&tv2, NULL);

      tv_usecs = (tv2.tv_sec - tv1.tv_sec) * 1E6 + (tv2.tv_usec - tv1.tv_usec);
#if ! COPY_PASTE_CALC
      printf("resultado = %d\t", resultado);
      printf("%s:%9ld us\n", msg, tv_usecs);
#else
      printf("%9ld us\n", tv_usecs);
#endif
}

int main() {
#if ! TEST
      int i; // inicializar array
      for (i = 0; i < SIZE; i++) // se queda en cache
            lista[i] = i;
#endif
```

```
        crono(paridad1, "Paridad1 (    en lenguaje C for  )");
        crono(paridad2, "Paridad2 (    en lenguaje C whi  )");
        crono(paridad3, "Paridad3 (Ejemplo CS:APP Ej: 3.22)");
        crono(paridad4, "Paridad4 (Traducción bucle While )");
        crono(paridad5, "Paridad5 (      Suma en árbol    )");
        crono(paridad6, "Paridad6 (Bucle interno con setpe)");
        exit(0);
}
```