PRACTICA 3

4.1.- Calcular la suma de bits de una lista de enteros sin signo

A continuación le adjunto el código de la primera parte de la practica. Este codigo se encuentra adjunto en un fichero llamado *“pop_count.c”*.

```
//  según la versión de gcc y opciones de optimización usadas, tal vez haga
falta
//  usar gcc –fno-omit-frame-pointer si gcc quitara el marco pila (%ebp)

#include <stdio.h>     // para printf()
#include <stdlib.h>    // para exit()
#include <sys/time.h>  // para gettimeofday(), struct timeval

#define SIZE (1<<22)   // tamaño suficiente para tiempo apreciable
#define WSIZE 8*sizeof(int)
int lista[SIZE];
int resultado=0;


int pcount_v1(int lista[], int size) {
  int result = 0;
  int x;
  for (int i = 0; i < size; i++) {
    x = lista[i];
    for (int j = 0; j < WSIZE; j++) {
    unsigned mask = 1 << j;
      result += ( x & mask) != 0;
    }
  }
  return result;
}

int pcount_v2(int lista[], int size) {
  int result = 0;
  int x,i=0;
  while (i < size) {
    x = lista[i];
    while(x){
      result += x & 0x1;
      x >>= 1;
    }
      i++;
  }
  return result;
}

int pcount_v3(int lista[], int size) {
  int result = 0;
  int x,i=0;
  while (i < size) {
    x = lista[i];
```

```
    asm("\n"
"bucle:           \n\t" //seguir mientras que x!=0
    "shr %[x]     \n\t" //desplaza un bit a la derecha x
    "adc $0,%[r]            \n\t" // Suma 0 a result + el acarreo
    "cmp $0,%[x]            \n\t" // Compara 0 con x
    "jne bucle            \n\t"  // Si no se cumple la comparacion anterior
vuelve a bucle
    : [r] "+r" (result)
    : [x] "r" (x)   );
      i++;
  }
  return result;
}

int pcount_v4(int lista[], int size) {
  int val, val2 = 0;
  int x;
  for (int i = 0; i < size; i++) {
    x = lista[i];
    val = 0;
    for (int j = 0; j < 8; j++) {
      val += x & 0x01010101;
      x >>= 1;
    }
    val += (val >> 16);
    val += (val >> 8);
    val = val & 0xFF;
    val2 += val;
  }
  return val2;
}

int pcount_v5(unsigned *array, int len)
{
  int i;
  unsigned val, result=0;
  int SSE_mask[] = {0x0f0f0f0f, 0x0f0f0f0f, 0x0f0f0f0f, 0x0f0f0f0f};
  int SSE_LUTb[] = {0x02010100, 0x03020201, 0x03020201, 0x04030302};
  //  3 2 1 0    7 6 5 4    1110 9 8   15141312

  if (len & 0x3)
    printf("leyendo 128b pero len no multiplo 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" // mascara
      "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" //; xmm2 += 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" //; xmm0 = [     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;
}

int pcount_v6(int lista[], int size) {
  int x, val, resultado = 0;
  for (int i = 0; i < size; i++) {
    x = lista[i];
    asm("popcnt %[x], %[val]        \n\t"
        : [val]"=r"(val)
        : [x] "r" (x)
    );
    resultado += val;
  }
  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);
    printf("resultado = %d\t", resultado);
    printf("%s:\t%9ld us\n", msg, tv_usecs);
}

int main()
{
    int i;             // inicializar array
    for (i=0; i<SIZE; i++)  // se queda en cache
          lista[i]=i;

    crono(pcount_v1, "pop_count_v1 (En c con for)");
    crono(pcount_v2, "pop_count_v2 (En c con while)");
```

```
    crono(pcount_v3, "pop_count_v3 (En c con asm)");
    crono(pcount_v4, "pop_count_v4 (Vers libro)");
    crono(pcount_v5, "pop_count_v5 (Con SSSE3)");
    crono(pcount_v6, "pop_count_v6 (Con SSSE4)");

    exit(0);
}
```

4.2.- Calcular la suma de paridades de una lista de enteros sin signo

A continuación le adjunto el código de la primera parte de la practica. Este codigo se encuentra adjunto en un fichero llamado *"parity.c"*.

```
//  según la versión de gcc y opciones de optimización usadas, tal vez haga
falta
//  usar gcc –fno-omit-frame-pointer si gcc quitara el marco pila (%ebp)

#include <stdio.h>     // para printf()
#include <stdlib.h>    // para exit()
#include <sys/time.h>  // para gettimeofday(), struct timeval

#define SIZE (1<<22)   // tamaño suficiente para tiempo apreciable
#define WSIZE 8*sizeof(int)
int lista[SIZE];
int resultado=0;

//La paridad devuelve 0 si el numero de 1nos es par y 1 si el numero de 1nos
es impar

int paridad_v1(int lista[], int size) {
 int result = 0, paridad = 0;
 int x;
 for (int i = 0; i < size; i++) {
  x = lista[i];
  for (int j = 0; j < WSIZE; j++) {
  unsigned mask = 1 << j;
   result ^= ( x & mask) != 0;
  }
  paridad += result;
 }
 return paridad;
}

int paridad_v2(int lista[], int size) {
 int result = 0, paridad = 0;
 int x,i=0;
 while (i < size) {
  x = lista[i];
```

```
  while(x){
   result ^= x & 0x1;
   x >>= 1;
  }
   paridad += result;
   i++;
 }
 return paridad;
}

int paridad_v3(int lista[], int size) {
 int val = 0, paridad = 0, i = 0, x;
 while(i < size){
  x = lista[i];
  while (x) {
   val ^= x;
   x >>= 1;
  }
  i++;
  paridad += val & 0x1;
 }
 return paridad;
}

int paridad_v4(int lista[], int size) {
 int val, paridad = 0, i = 0, x;
 while(i < size){
  x = lista[i];
  val = 0;
  asm("\n"
"bucle:                  \n\t" // seguir mientras que x!=0
  "xor %[x],%[v]       \n\t" // operacion xor de x y val
  "shr %[x]            \n\t" // desplaza un bit a la derecha x
  "cmp $0,%[x]         \n\t" // compara si x vale 0
  "jne bucle           \n\t" // si no vale 0 vuelve al bucle
  : [v] "+r" (val)   // salida de val
  : [x] "r" (x)    // entrada de x
  );
  i++;
  paridad += val & 0x1;
 }
 return paridad;
}

int paridad_v5(int lista[], int size) {
 int x, paridad = 0;
 for (int i = 0; i < size; i++) {
  x = lista[i];
  for (int j = 16; j >= 1; j=j/2) {
   x ^= (x >> j);
  }
  x  = x & 0x1;
  paridad += x;
 }
```

```
 return paridad;
}

int paridad_v6(int lista[], int size) {
 int x, paridad = 0;
 for (int i = 0; i < size; i++) {
  x = lista[i];
  for (int j = 16; j >= 1; j=j/2) {
   // x ^= (x >> j);
   asm("\n"
   "mov %[x], %%edx                  \n\t"    // mueve el valor de x a el
registro %edx
   "shr $16, %[x]                       \n\t" // desplaza x 16 bits a la
derecha
   "xor %[x], %%edx                  \n\t" // realiza la operacion xor entre
x y el contenido del registro %edx
   "xor %%dh, %%dl                      \n\t" // realiza la operacion xor
entre %dh y %dl
   "setpo %%dl                         \n\t" // mira el flag PF que señala la
paridad. Si PF esta clear pasa 1 a %dl y si no pasa 0 a %dl
   "movzx %%dl, %[x]                \n\t" // devuelve el valor en 32bits
   : [x]"+r" (x)      // e/s: entrada con valor de x, y salida con paridad
   :
   :"edx"                                          // clobber
   );
  }
  x = x & 0x1;
  paridad += x;
 }
 return paridad;
}

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);
  printf("resultado = %d\t", resultado);
  printf("%s:\t%9ld us\n", msg, tv_usecs);
}

int main()
{
  int i;                // inicializar array
  for (i=0; i<SIZE; i++) // se queda en cache
        lista[i]=i;

  crono(paridad_v1, "paridad_v1 (En c con for)");
  crono(paridad_v2, "paridad_v2 (En c con while)");
  crono(paridad_v3, "paridad_v3 (Vers libro)");
```

```
  crono(paridad_v4, "paridad_v4 (En c con asm)");
  crono(paridad_v5, "paridad_v5 (Vers libro)");
  crono(paridad_v6, "paridad_v6 (En c con asm)");

  exit(0);
}
```

4.3.- Gráficas

Para una mejor visión de las graficas y los datos obtenidos, le adjunto dos ficheros Excel llamados *"Parity.xlsx"* y *"Pop_count.xlsx"*.