Práctica 3

Alejandro Poyatos López

2ºB(2)

En la primera clase de prácticas correspondiente a esta sesión estuve haciendo los ejercicios de las sumas correspondientes al tutorial, durante el resto de las clases he estado trabajando con el código de parity.c y popcount.c, y finalmente los he compilado con distintos ordenes de optimización para recopilar los datos que plasmaré en la gráfica de esta práctica.

Codigo parity.c:

#include <stdio.h>

#include <stdlib.h>

#include <sys/time.h>

#define WSIZE 8\*sizeof(unsigned)

#define SIZE 1<<20

unsigned lista[SIZE];

int resultado = 0;

int paridad1(int\* array, int len){

int resultado = 0;

int x;

int i = 0;

unsigned val = 0;

for (i = 0; i < len; i++){

x = array[i];

do{

val ^= (x & 0x1);

}while(x >>= 1);

resultado += val;

val = 0;

}

return resultado;

}

int paridad2(int\* array, int len){

int resultado = 0;

int i;

int j;

unsigned x;

unsigned val = 0;

for(i = 0; i < len; i++){

x = array[i];

for(j = 0; j < WSIZE; j++){

unsigned mask = 0x1 << j;

val ^= (x & mask) != 0;

}

resultado += val;

val = 0;

}

return resultado;

}

int paridad3(int\* array, int len){

int i;

int x;

int resultado = 0;

int val = 0;

for (i = 0; i < len; i++){

x = array[i];

while(x){

val ^= x;

x >>= 1;

}

resultado += (val & 0x1);

}

return resultado;

}

int paridad4(int\* array, int len){

unsigned result = 0, i, val, x;

for (i = 0; i < len; i++){

x = array[i];

val = 0;

asm("\n"

"bucle: \n\t"

"xor %[x], %[val] \n\t" //val ^= x

"shr $1, %[x] \n\t"

"jnz bucle \n\t"

: [val] "+r" (val)

: [x] "r" (x)

);

result += (val & 1);

}

return result;

}

int paridad5(unsigned\* array, int len){

int i, j, x;

int result = 0;

for(i = 0; i < len; i++){

x = array[i];

for(j = 16; j >= 1; j = j / 2){

x ^= x >> j;

}

result += (x & 0x1);

}

return result;

}

int paridad6(unsigned\* array, int len){

int i;

unsigned x;

int result = 0;

for (i = 0; i < len; i++){

x=array[i];

asm("\n"

"mov %[x], %%edx \n\t" //sacar copia para XOR. Controlar el registro…

"shr $16, %[x] \n\t" //x >>= 1

"xor %[x], %%edx \n\t"

"xor %%dh, %%dl \n\t"

"setpo %%dl \n\t"

"movzx %%dl, %[x] \n\t" //devolver en 32 bits

: [x] "+r" (x) //Entra valor elemento, sale paridad

:

: "edx"

);

result += x;

}

return result;

}

void crono(int (\*func)(), char\* msg){ //Funcion usada para medir los tiempos

struct timeval tv1,tv2;

long tv\_usecs;

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:%9ld us\n", msg, tv\_usecs);

}

int main(){

int i;

for (i = 0; i < SIZE; i++)

lista[i] = i;

crono(paridad1, "paridad1");

crono(paridad2, "paridad2");

crono(paridad3, "paridad3");

crono(paridad4, "paridad4");

crono(paridad5, "paridad5");

crono(paridad6, "paridad6");

exit(0);

}

Código popcount.c:

#include <stdio.h>

#include <stdlib.h>

#include <sys/time.h>

#define WSIZE 8\*sizeof(unsigned)

#define SIZE 1<<20

unsigned lista[SIZE];

int resultado = 0;

int popcount1(int\* array, int len){

int resultado = 0;

int i = 0;

int x;

for (i = 0; i < len; i++){

x = array[i];

do{

resultado += x & 0x1;

x >>= 1;

}while(x);

}

return resultado;

}

int popcount2(int\* array, int len){

int resultado = 0;

int i, j;

unsigned x;

for(i = 0; i < len; i++){

x = array[i];

for(j = 0; j < WSIZE; j++){

unsigned mask = 1 << j;

if((x & mask)){

resultado++;

}

}

}

return resultado;

}

int popcount3(int\* array, int len){

int i, x;

int resultado = 0;

for (i = 0; i < len; i++){

x = array[i];

asm("\n"

"ini3: \n\t" //seguir mientras que x!=0

"shr %[x] \n\t" //LSB en CF

"adc $0x0, %[r] \n\t"

"test %[x], %[x] \n\t"

"jnz ini3 \n\t" //Salto si != 0

: [r] "+r" (resultado) // e/s: añadir lo acumulado hasta el momento

: [x] "r" (x) //entrada: valor elemento

);

}

return resultado;

}

int popcount4(int\* array, int len){

int result = 0, val, i, j;

unsigned x;

for (i = 0; i < len; i++){

val = 0;

x = array[i];

for (j = 0;j < 8; j++){

val += x & 0x1010101;

x >>= 1;

}

val += (val >> 16);

val += (val >> 8);

result += (val & 0xFF);

}

return result;

}

int popcount5(unsigned\* array, int len){

int i, val=0, result=0;

int SSE\_mask[] = {0x0f0f0f0f, 0x0f0f0f0f, 0x0f0f0f0f, 0x0f0f0f0f};

int SSE\_LUTb[] = {0x02010100, 0x03020201, 0x03020201, 0x04030302};

if (len & 0x3){

printf("Leyendo 128 b, pero ¿len no es multiplo de 4?\n");

}

for (i = 0; i < len; i += 4){

asm (

"movdqu %[x], %%xmm0 \n\t"

"movdqa %%xmm0, %%xmm1 \n\t"

"movdqu %[m], %%xmm6 \n\t"

"psrlw $4 , %%xmm1 \n\t"

"pand %%xmm6, %%xmm0 \n\t"

"pand %%xmm6, %%xmm1 \n\t"

"movdqu %[l], %%xmm2 \n\t"

"movdqa %%xmm2, %%xmm3 \n\t"

"pshufb %%xmm0, %%xmm2 \n\t"

"pshufb %%xmm1, %%xmm3 \n\t"

"paddb %%xmm2, %%xmm3 \n\t"

"pxor %%xmm0, %%xmm0 \n\t"

"psadbw %%xmm0, %%xmm3 \n\t"

"movhlps %%xmm3, %%xmm0 \n\t"

"paddd %%xmm3, %%xmm0 \n\t"

"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 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]"

: [val] "=r" (val)

: [x] "r" (x)

);

result+=val;

}

return result;

}

void crono(int (\*func)(), char\* msg){

struct timeval tv1,tv2;

long tv\_usecs;

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:%9ld us\n", msg, tv\_usecs);

}

int main(){

for (int i = 0; i < SIZE; i++)

lista[i] = i;

crono(popcount1, "popcount1");

crono(popcount2, "popcount2");

crono(popcount3, "popcount3");

crono(popcount4, "popcount4");

crono(popcount5, "popcount5");

crono(popcount6, "popcount6");

exit(0);

}

En cuanto las gráficas se han realizado en Excel y los programas se han ejecutado en la terminal de mi Ubuntu 16.04 ejecutándose sobre una máquina virtual, en cuanto a las prestaciones del equipo se ha realizado sobre un Intel Core i7 7700HQ:



