Segunda práctica

Estructura de computadores

Escuela Técnica Superior de Ingeniería Informática de Granada (ETSIIT)

Juan Pablo Sáez Gutiérrez
2º Ingeniería Informática (1º Cuatrimestre)
Granada - 4 de noviembre de 2018

ETSIIT
Escuela Técnica Superior de Ingenierías Informática y de Telecomunicación

Índice general

1. The population count **2**
1.1. Implementaciones Popcount 2

Capítulo 1

Popcount

Popcount, tambien llamado peso Hamming es el número de bits a 1 en una palabra de n bits. Este cálculo se puede implementar de diversas formas. Para este caso concreto se han escrito diez formas de hacer el cálculo de popcount, con el objetivo de comprobar cual de ellas tiene una mayor eficiencia.

Se trata por tanto de programar varias versiones de una función que sume los pesos Hamming (no de bits activados) de todos los elementos de un array, con y sin ensamblador en-línea, y comprobando siempre la corrección del resultado calculado.

1.1. Versiones Popcount 1....10

Incluidas junto a este documento, en el archivo popcount.c se encuentran las 10 versiones. En script.sh se encuentra el script proporcionado en los materiales que ha sido utilizado para conseguir los tiempos y valores de las funciones popcount.

lscpu:	CPU(s): 8 Nombre del modelo: Intel(R) Core(TM) i5-8250U CPU @ 1.60GHz Virtualización: VT-x Caché L3: 6144K

POPCOUNT:	for i in 0 g 1 2; do
	printf "__OPTIM%1c__%48s\n" $i "" | tr " " "="
	rm popcount
	gcc popcount.c -o popcount -O$i -D TEST=0
	for j in $(seq 0 10); do
	echo $j; ./popcount
	done | pr -11 -l 22 -w 80
	done
	
	ignorar medición 0, repetir columna si alguna medición se sale demasiado de la media

Optimización -O0	0	1	2	3	4	5	6	7	8	9	10	media
popcount1 (lenguaje C - for):	73557	73695	73549	73899	73508	73528	73505	73578	73548	73508	73871	73619
popcount2 (lenguaje C - while):	38900	40956	40620	38819	38986	41645	40474	41403	39676	39033	40146	40176
popcount3 (leng.ASM-body while 4i):	11225	11176	11209	11208	11202	11191	11193	11205	11215	11201	11060	11186
popcount4 (leng.ASM-body while 3i):	10288	10303	10287	10317	10307	10308	10319	10309	10296	10298	10121	10287
popcount5 (CS:APP2e 3.49-group 8b):	19272	19267	19269	19266	19265	19263	19267	19270	19275	19263	19267	19267
popcount6 (Wikipedia- naive - 32b):	7279	7268	7265	7262	7258	7261	7262	7265	7269	7262	7084	7246
popcount7 (Wikipedia- naive -128b):	5453	5453	5453	5447	5451	5451	5452	5451	5456	5451	5315	5438
popcount8 (asm SSE3 - pshufb 128b):	708	697	697	689	690	690	708	693	692	736	675	697
popcount9 (asm SSE4- popcount 32b):	2571	2558	2569	2570	2565	2565	2564	2576	2565	2569	2559	2566
popcount10(asm SSE4- popcount128b):	1193	1194	1194	1188	1188	1189	1190	1189	1192	1193	1189	1191

Optimización -Og	0	1	2	3	4	5	6	7	8	9	10	media
popcount1 (lenguaje C - for):	20438	20462	20445	20489	20432	20439	20436	20609	20430	20434	20415	20459
popcount2 (lenguaje C - while):	8124	10869	8237	8354	8191	8207	8128	8239	8126	8173	8150	8467
popcount3 (leng.ASM-body while 4i):	10076	10119	10148	10287	10123	10063	10182	10122	10147	10077	10060	10133
popcount4 (leng.ASM-body while 3i):	9521	9525	9482	9587	9512	9521	9517	9529	9512	9516	9511	9521
popcount5 (CS:APP2e 3.49-group 8b):	4981	4959	4993	5031	4957	4957	4965	4958	4959	4957	4958	4969
popcount6 (Wikipedia- naive - 32b):	3063	2560	2621	2650	2576	2560	2559	2568	2585	2558	2560	2580
popcount7 (Wikipedia- naive -128b):	2220	2173	2233	2276	2172	2162	2164	2172	2162	2161	2164	2184
popcount8 (asm SSE3 - pshufb 128b):	403	377	403	411	382	375	386	374	379	373	370	383
popcount9 (asm SSE4- popcount 32b):	675	621	635	623	621	620	620	619	620	620	620	622
popcount10(asm SSE4- popcount128b):	871	317	347	350	318	316	319	317	317	316	316	323

Optimización -O1	0	1	2	3	4	5	6	7	8	9	10	media
popcount1 (lenguaje C - for):	12869	13005	12934	12867	12854	12865	12868	12864	12862	12880	12890	12889
popcount2 (lenguaje C - while):	8165	8350	8245	8180	8193	8205	8195	8202	8171	8158	8214	8211
popcount3 (leng.ASM-body while 4i):	12382	12388	12662	12377	12378	12377	12379	12378	12378	12382	12380	12408
popcount4 (leng.ASM-body while 3i):	9460	9485	9583	9444	9437	9460	9466	9437	9449	9456	9432	9465
popcount5 (CS:APP2e 3.49-group 8b):	5263	5260	5345	5257	5259	5261	5262	5260	5263	5260	5260	5269
popcount6 (Wikipedia- naive - 32b):	2559	2559	2697	2559	2559	2558	2559	2579	2560	2560	2560	2575
popcount7 (Wikipedia- naive -128b):	2069	2070	2207	2069	2078	2068	2069	2078	2068	2070	2080	2086
popcount8 (asm SSE3 - pshufb 128b):	379	379	421	374	371	381	373	381	374	379	373	381
popcount9 (asm SSE4- popcount 32b):	452	450	499	450	449	453	449	455	448	448	450	455
popcount10(asm SSE4- popcount128b):	324	317	355	318	316	322	317	324	316	318	319	322

Optimización -O2	0	1	2	3	4	5	6	7	8	9	10	media
popcount1 (lenguaje C - for):	12893	12872	12857	12853	12794	12865	12776	12782	12853	12786	12788	12823
popcount2 (lenguaje C - while):	7890	8002	7970	7955	7838	8009	7811	7854	7947	7808	7815	7901
popcount3 (leng.ASM-body while 4i):	10011	10047	9986	9972	10016	10086	9998	9982	9997	10011	9973	10007
popcount4 (leng.ASM-body while 3i):	9492	9531	9478	9462	9534	9539	9487	9496	9449	9485	9483	9494
popcount5 (CS:APP2e 3.49-group 8b):	4179	4226	4185	4217	4359	4219	4178	4179	4213	4183	4177	4214
popcount6 (Wikipedia- naive - 32b):	2200	2281	2190	2235	2219	2283	2191	2190	2265	2190	2190	2223
popcount7 (Wikipedia- naive -128b):	2000	2091	2003	2081	2007	2108	1999	2002	2145	1999	1999	2043
popcount8 (asm SSE3 - pshufb 128b):	374	378	383	409	400	419	375	377	497	379	377	399
popcount9 (asm SSE4- popcount 32b):	455	454	458	518	461	515	454	453	591	469	456	483
popcount10(asm SSE4- popcount128b):	323	317	328	344	329	351	318	317	380	325	317	333

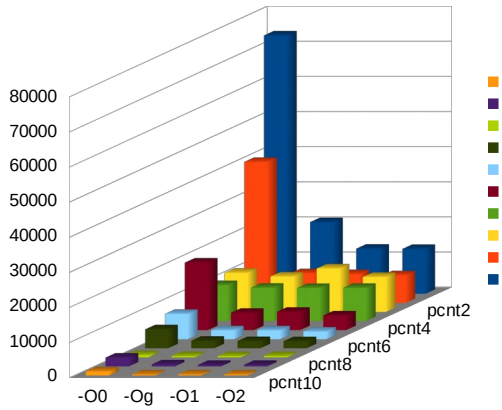
POPCOUNT:	-O0	-Og	-O1	-O2
pcnt1	73619	20459	12889	12823
pcnt2	40176	8467	8211	7901
pcnt3	11186	10133	12408	10007
pcnt4	10287	9521	9465	9494
pcnt5	19267	4969	5269	4214
pcnt6	7246	2580	2575	2223
pcnt7	5438	2184	2086	2043
pcnt8	697	383	381	399
pcnt9	2566	622	455	483
pcnt10	1191	323	322	333

Ganancias:	-O0	-Og	-O1	-O2
pcnt1		0,63	1,00	1,01
pcnt2		1,52	1,57	1,63
pcnt3		1,27	1,04	1,29
pcnt4		1,35	1,36	1,36
pcnt5		2,59	2,45	3,06
pcnt6		5,00	5,01	5,80
pcnt7		5,90	6,18	6,31
pcnt8		33,65	33,86	32,27
pcnt9		20,73	28,32	26,69
pcnt10		39,87	40,00	38,75

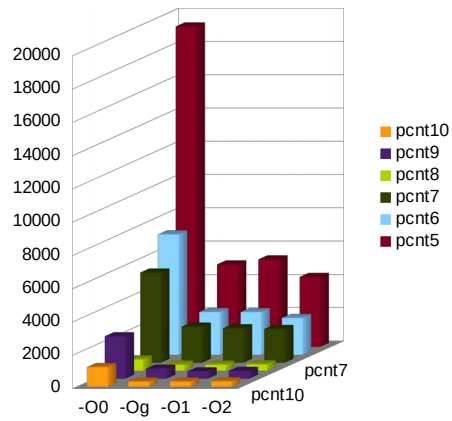
Comentario

comparado con el for más rápido
el while es un 63% más rápido
ASM se queda en un 29%
o en un 36%
sumar en grupos 8b sale 3x más rápido
sumar en árbol 5.8x
lectura 128b sube a 6.3x
SSSE3 sube a 33.86 más rápido
SSE4 sólo 28.32x por leer 32b
SSE4 128b sube a 40x

bucles for/while



sumas en árbol



repertorio multimedia