lscpu:	CPU(s): 2 Nombre del modelo: Intel(R) Core(TM) Virtualización: VT-x Caché L3: 6144K

POPCOUNT:

```
for i in 0 g 1 2; do
    printf "__OPTIM%1c__%48s\n" $1 "" | tr " " "="
    rm  popcount
    gcc popcount.c -o popcount -O$1 -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 medici

Optimización -O0	0	1	2	3	4
popcount1 (lenguaje C - for):	72945	70626	69377	68034	69345
popcount2 (lenguaje C - while):	47842	43719	52397	41543	41973
popcount3 (leng.ASM-body while 41):	13208	12712	12614	13471	12479
popcount4 (leng.ASM-body while 31):	11276	10242	10831	10163	10283
popcount5 (CS:APP2e 3.49-group 8b):	19268	18648	18742	18482	18724
popcount6 (Wikipedia- naive - 32b):	8080	8003	7973	7890	7904
popcount7 (Wikipedia- naive -128b):	5809	5821	5677	5906	5704
popcount8 (asm SSE3 - pshufb 128b):	812	817	753	772	743
popcount9 (asm SSE4- popcount 32b):	2592	2691	2509	3208	2515
popcount10(asm SSE4- popcount128b):	1243	1272	1196	1247	1188

Optimización -Og	0	1	2	3	4
popcount1 (lenguaje C - for):	19868	19448	19503	19583	19601
popcount2 (lenguaje C - while):	8777	8621	8609	8710	8611
popcount3 (leng.ASM-body while 41):	10653	10707	10692	10772	10725
popcount4 (leng.ASM-body while 31):	10289	8893	8801	9222	8926
popcount5 (CS:APP2e 3.49-group 8b):	6188	5738	5830	5798	5787
popcount6 (Wikipedia- naive - 32b):	2840	2665	2603	2573	2699
popcount7 (Wikipedia- naive -128b):	2223	2209	2142	2179	2258
popcount8 (asm SSE3 - pshufb 128b):	412	420	396	379	427
popcount9 (asm SSE4- popcount 32b):	493	466	420	430	491
popcount10(asm SSE4- popcount128b):	341	342	313	310	366

Optimización -O1	0	1	2	3	4
popcount1 (lenguaje C - for):	14745	14721	14867	14517	14605
popcount2 (lenguaje C - while):	10077	9904	9819	12247	10957
popcount3 (leng.ASM-body while 41):	12868	10679	10623	10822	10611
popcount4 (leng.ASM-body while 31):	9390	9371	8864	8969	8894
popcount5 (CS:APP2e 3.49-group 8b):	5488	5436	5540	5590	5493
popcount6 (Wikipedia- naive - 32b):	2722	2739	2723	2706	2760
popcount7 (Wikipedia- naive -128b):	2128	2123	2144	2215	2118
popcount8 (asm SSE3 - pshufb 128b):	417	394	399	423	418
popcount9 (asm SSE4- popcount 32b):	468	445	444	493	464
popcount10(asm SSE4- popcount128b):	331	331	325	432	333

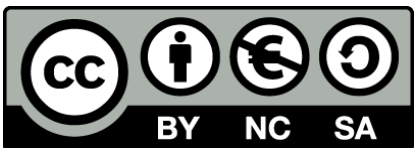
Optimización -O2	0	1	2	3	4
popcount1 (lenguaje C - for):	19192	17314	16927	16699	18175
popcount2 (lenguaje C - while):	8081	7665	7672	7762	8008
popcount3 (leng.ASM-body while 41):	10320	10452	10367	12119	10560
popcount4 (leng.ASM-body while 31):	8836	9198	8813	12338	8910
popcount5 (CS:APP2e 3.49-group 8b):	4072	4966	4967	4729	5132
popcount6 (Wikipedia- naive - 32b):	2138	2116	2262	2912	2117
popcount7 (Wikipedia- naive -128b):	1930	1935	1946	2386	1941
popcount8 (asm SSE3 - pshufb 128b):	355	355	359	473	355
popcount9 (asm SSE4- popcount 32b):	417	445	417	550	417
popcount10(asm SSE4- popcount128b):	284	285	285	399	284

POPCOUNT:	-O0	-Og	-O1	-O2
pcnt1	69313	19448	**14499**	17063
pcnt2	42929	**8603**	10149	7968

i7-4870HQ CPU @2.5 GHz

"

ón se sale demasiado de la media



Prácticas de Estructura de Computadores
por Javier Fernández y Mancia Anguita

Zona para reproducir r
recordar que se ignora

5	6	7	8	9	10	media
68952	69793	72471	68164	67964	68399	69313
42125	41725	39988	42113	41899	41808	42929
12548	12467	12739	12715	12643	12476	12686
10330	10252	10354	10172	10536	10113	10328
18610	18492	18562	18474	18706	19636	18708
7988	7899	8000	7900	7918	8187	7966
5796	5710	5812	5664	5694	6542	5833
851	807	819	743	752	763	782
2612	2556	2563	2513	2509	2533	2621
1272	1186	1248	1195	1166	1168	1214

media	0	1
#¡DIV/0!		
#¡DIV/0!		
#¡DIV/0!		
#¡DIV/0!		
#¡DIV/0!		
#¡DIV/0!		
#¡DIV/0!		
#¡DIV/0!		
#¡DIV/0!		
#¡DIV/0!		

5	6	7	8	9	10	media
19376	19356	19309	19431	19487	19381	19448
8609	8512	8580	8768	8493	8517	8603
10672	10641	10720	11481	10741	10613	10776
9015	9495	8903	10204	8845	8818	9112
5760	5740	5742	5910	5808	5776	5789
2668	2664	2658	2781	2651	2662	2662
2203	2245	2207	2216	2216	2184	2206
415	403	419	400	410	393	406
458	453	457	557	483	471	469
334	336	331	345	352	346	338

media	0	1
#¡DIV/0!		
#¡DIV/0!		
#¡DIV/0!		
#¡DIV/0!		
#¡DIV/0!		
#¡DIV/0!		
#¡DIV/0!		
#¡DIV/0!		
#¡DIV/0!		
#¡DIV/0!		

5	6	7	8	9	10	media
14680	14769	14220	14271	14023	14317	14499
9386	11330	9902	9248	8717	9983	10149
10593	11331	10686	10741	10535	10542	10716
9036	11332	8830	8769	8901	8824	9179
5477	11333	5342	5440	5512	5459	6062
2672	11334	2673	2625	2678	2656	3557
2124	11335	2123	2001	2365	2122	3067
393	11336	392	406	483	389	1503
443	11337	440	417	470	462	1542
326	11338	325	288	330	329	1436

media	0	1
#¡DIV/0!		
#¡DIV/0!		
#¡DIV/0!		
#¡DIV/0!		
#¡DIV/0!		
#¡DIV/0!		
#¡DIV/0!		
#¡DIV/0!		
#¡DIV/0!		
#¡DIV/0!		

5	6	7	8	9	10	media
17020	16915	16869	16841	16808	17057	17063
9614	7701	7730	7790	7666	8075	7968
10995	10505	10458	10560	10386	10583	10699
10735	8883	8925	8832	8794	13239	9867
4925	4426	4424	4555	4533	4181	4684
2251	2197	2209	2197	2202	2205	2267
2100	1975	1975	1973	1984	1977	2019
609	382	378	379	380	380	405
592	445	438	441	439	442	463
362	316	312	312	333	325	321

media	0	1
#¡DIV/0!		
#¡DIV/0!		
#¡DIV/0!		
#¡DIV/0!		
#¡DIV/0!		
#¡DIV/0!		
#¡DIV/0!		
#¡DIV/0!		
#¡DIV/0!		
#¡DIV/0!		

	-O0	-Og	-O1	-O2
pcnt1			1,00	
pcnt2		**1,69**		

Comentario

comparado con el for más rápido
el while es un 70% más rápido

mediciones
medición 0

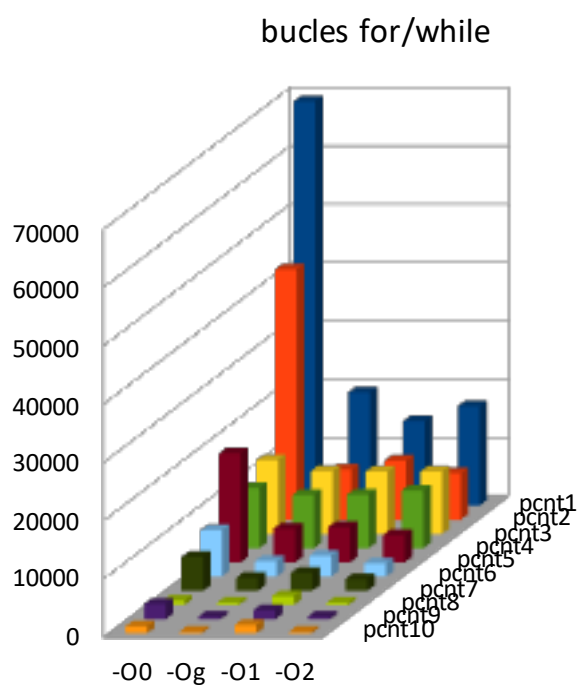
2	3	4	5	6	7	8	9	10

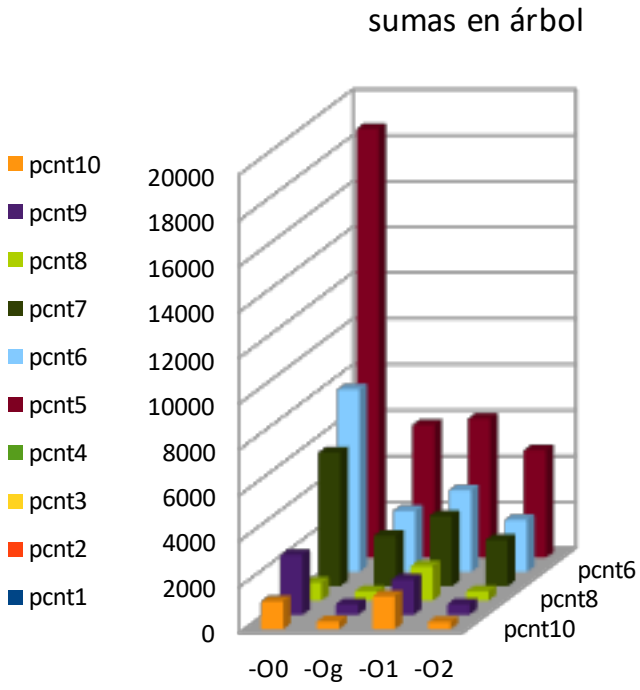
2	3	4	5	6	7	8	9	10

2	3	4	5	6	7	8	9	10

2	3	4	5	6	7	8	9	10

pcnt3	12686	10776	**10716**	10699
pcnt4	10328	9112	**9179**	9867
pcnt5	18708	5789	6062	**4684**
pcnt6	7966	2662	3557	**2267**
pcnt7	5833	2206	3067	**2019**
pcnt8	782	406	1503	**405**
pcnt9	2621	469	1542	**463**
pcnt10	1214	**338**	**1436**	**321**





pcnt3		1,35	
pcnt4		1,58	
pcnt5			**3,10**
pcnt6			**6,40**
pcnt7			**7,18**
pcnt8			**35,80**
pcnt9			31,34
pcnt10	**42,96**	10,10	45,13

ASM se queda en un 35%
o en un 43%
sumar en grupos 8b sale 3x más rápido
sumar en árbol 6x
lectura 128b sube a 10x
SSSE3 sube a 35x más rápido
SSE4 sólo 30x por leer 32b
SSE4 128b sube a 44x