CMPEN 431 Project 2

A. Burak Gulhan 920177802
Agasthya Harekal 915750011

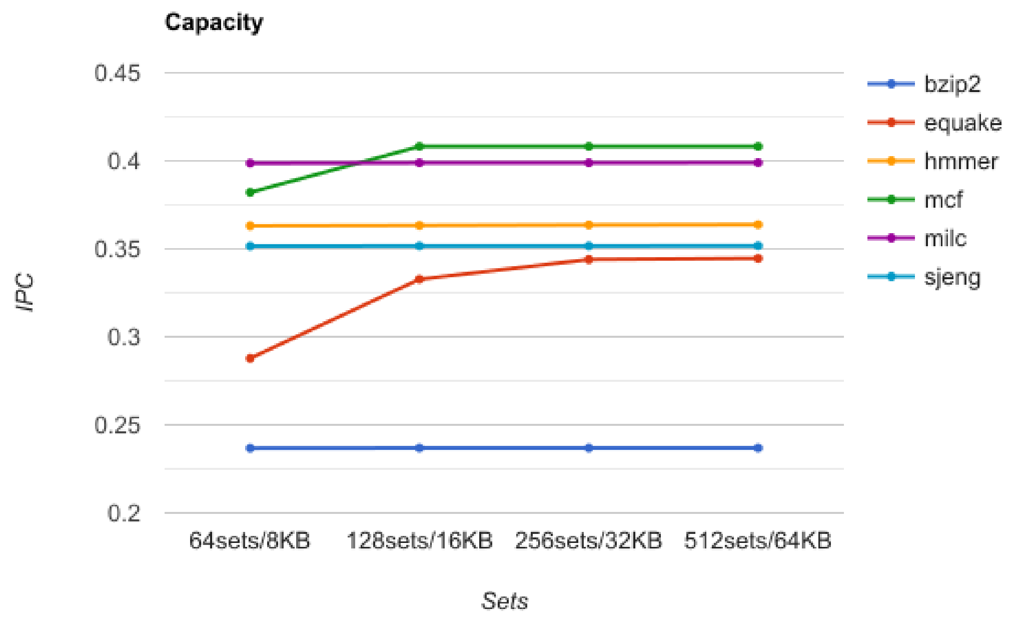
In tmp.cfg our latencies for dl1, dl2 and il2 were set as follows (using the latency calculation given in project 1):

-cache:dl1lat 3
-cache:dl2lat 8
-cache:il2lat 8

These 3 latency values are the same for all tests.
The il1 latency values were also calculated using the calculations given in project 1.

Capacity:

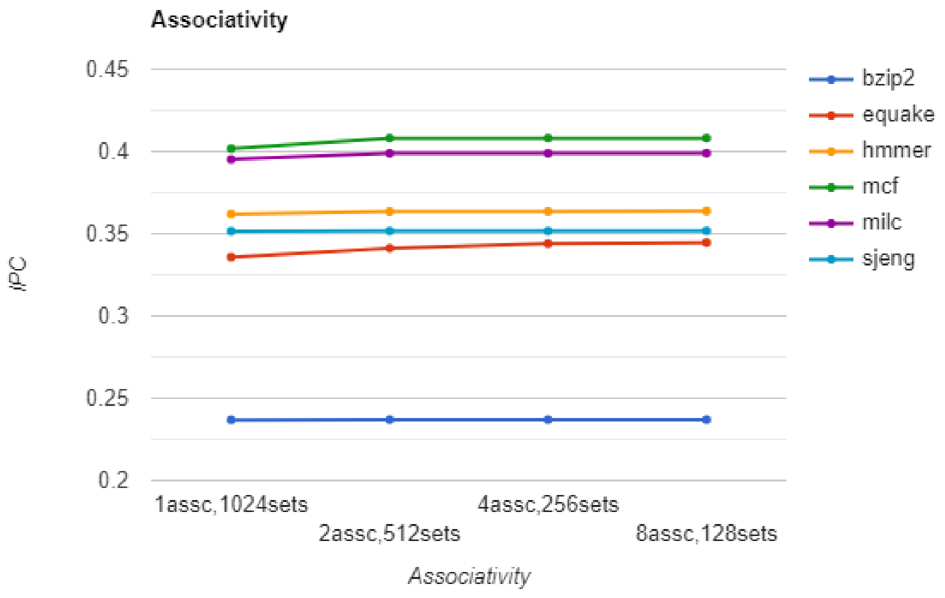
capacity	bzip2	equake	hmmer	mcf	milc	sjeng
64 sets / 8KB	0.2368	0.2878	0.3631	0.3821	0.3987	0.3516
128 sets / 16KB	0.2369	0.3328	0.3633	0.4082	0.3989	0.3517
256 sets / 32KB	0.2369	0.3440	0.3636	0.4082	0.3989	0.3517
512 sets / 64KB	0.2369	0.3446	0.3638	0.4082	0.3990	0.3518



Increasing cache helps take advantage of temporal locality up to a certain point. This point is the working set size. Up to this point the hit rate (and therefore IPC) increases quickly, but after this point it slows down considerably. Also increasing cache size increases latency, so increasing the cache size past the working set size can decrease IPC.
In the graph, we can see equake and mcf have a larger working memory size then the other benchmarks, since these two show a large increase in IPC, whereas the other benchmarks' IPC barely change.

Associativity:

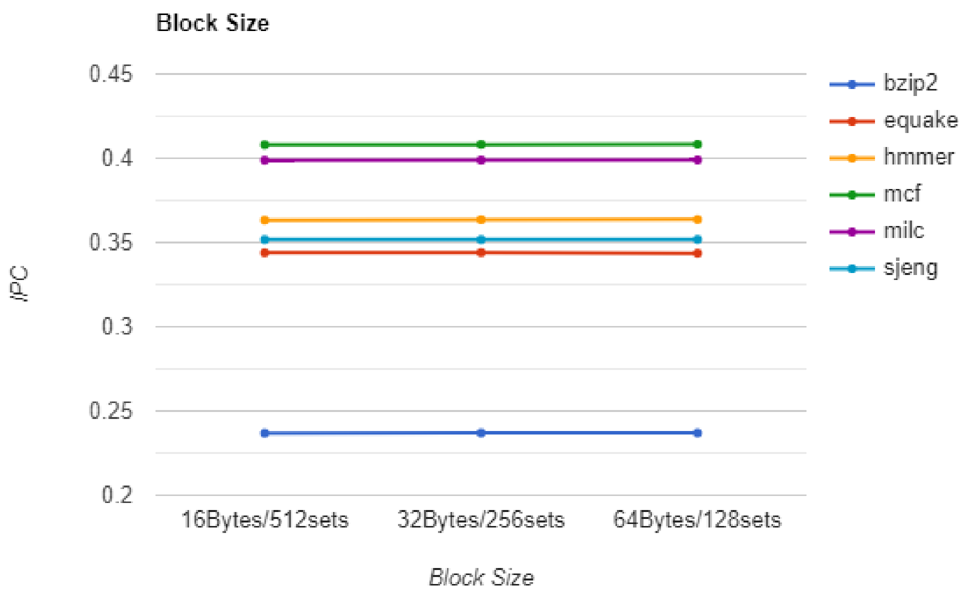
associativity	bzip2	equake	hmmer	mcf	milc	sjeng
1 associativity, 1024 sets	0.2368	0.3358	0.3620	0.4018	0.3952	0.3516
2 associativity, 512 sets	0.2369	0.3412	0.3635	0.4082	0.3989	0.3517
4 associativity, 256 sets	0.2369	0.3440	0.3636	0.4082	0.3989	0.3517
8 associativity, 128 sets	0.2369	0.3446	0.3638	0.4082	0.3990	0.3518



Increasing associativity helps decrease conflict misses, but also increases cache latency. From the graph we can see that equake, mcf and to a smaller extent hmmer gain IPC from increasing associativity. So we can assume that these 3 benchmarks have a higher number of conflict misses compared to the other benchmarks, since these 3 benchmarks benefit the most from increasing associativity.

Block Size:

block size	bzip2	equake	hmmer	mcf	milc	sjeng
16 Bytes / 512 sets	0.2368	0.3439	0.3633	0.4080	0.3988	0.3517
32 Bytes / 256 sets	0.2369	0.3440	0.3636	0.4082	0.3989	0.3517
64 Bytes / 128 sets	0.2369	0.3436	0.3637	0.4083	0.3990	0.3518



From the graph, it seems like there isn't any change in IPC from increasing block size. However, when we look at the table we see that all benchmarks except for equake increase in IPC, whereas equake decreases in IPC at 64B block size.

Increasing block size helps take advantage of spatial locality, but if the program does not have much spatial locality increasing block size doesn't give much of a benefit after a certain point and can even decrease IPC since larger blocks can displace cache items that might have been accessed later, such as the case with equake. We can assume that a block size of 16B is enough to take advantage of almost all spatial locality of these benchmarks, since IPC barely changes.