CSC369 A3 Task 3

By: Eric Koehli

Student #: 1004373974

Part 1: traceprogs

./sim -f traceprogs/tr-simpleloop.ref -m 50 -s 2635 -a <algorithm>

Algorithm	Hit Rate	Hit Count	Miss Count	Overall Eviction Count	Clean Eviction Count	Dirty Eviction Count
rand	70.85	7249	2983	2933	133	2800
fifo	71.05	7270	2962	2912	125	2787
clock	72.44	7412	2820	2770	78	2692
lru	72.82	7451	2781	2731	67	2664

./sim -f traceprogs/tr-simpleloop.ref -m 100 -s 2635 -a <algorithm>

Algorithm	Hit Rate	Hit Count	Miss Count	Overall Eviction Count	Clean Eviction Count	Dirty Eviction Count
rand	72.99	7468	2764	2664	41	2623
fifo	73.05	7474	2758	2658	34	2624
clock	73.58	7529	2703	2603	17	2586
lru	73.77	7548	2684	2584	1	2583

./sim -f traceprogs/tr-matmul.ref -m 50 -s 1093 -a <algorithm>

Algorithm	Hit Rate	Hit Count	Miss Count	Overall Eviction Count	Clean Eviction Count	Dirty Eviction Count
rand	65.54	1892630	995242	995192	478096	517096
fifo	60.97	1760605	1127267	1127217	541682	585535
clock	65.74	1898469	989403	989353	490968	498385
lru	63.94	1846647	1041225	1041175	520102	521073

./sim -f traceprogs/tr-matmul.ref -m 100 -s 1093 -a <algorithm>

Algorithm	Hit Rate	Hit Count	Miss Count	Overall Eviction Count	Clean Eviction Count	Dirty Eviction Count
rand	88.82	2564911	322961	322861	157894	164967
fifo	62.48	1804326	1083546	1083446	530672	552774
clock	89.44	2582850	305022	304922	151701	153221
lru	65.15	1881418	1006454	1006354	502791	503563

Part 2: trace{1, 2, 3}

./sim -f trace1 -m 8 -s 16 -a <algorithm>

Algorithm	Hit Rate	Hit Count	Miss Count
fifo	38.30	18	29
clock	40.43	19	28
lru	42.55	20	27

./sim -f trace2 -m 8 -s 16 -a <algorithm>

Algorithm	Hit Rate	Hit Count	Miss Count
fifo	84.00	42	8
clock	84.00	42	8
lru	84.00	42	8

./sim -f trace3 -m 8 -s 16 -a <algorithm>

Algorithm	Hit Rate	Hit Count	Miss Count
fifo	0.00	0	50
clock	58.00	29	21
lru	0.00	0	50

Part 3: Write up

In Part1, the results from my traceprogs simulations make sense. Within each group of the same memory size (e.g. m = 50), we can see that the hit rate increases, albeit not very much because the memory size is small. This trend makes sense based on the performance of the different algorithms; LRU should theoretically have the best hit rate, while Clock should be a close second, followed by FIFO, and random, as is observed for simpleloop.ref. However, for matmul.ref, this trend of hit rate based on the algorithm does not hold because (I am presuming based off the textbook) matmul is a matrix multiplication program, and so the memory references are somewhat scattered and therefore do not benefit from the “principles of locality” as described by the textbook.

On the other hand, the second trend that we should expect to see across all tables is that the hit rate should increase as the memory size increases. This is consistently observed, and intuitively makes sense because if the size of our memory increase, then of course we are going to see more hits since more pages are held in physical memory. Another note that is not shown in the tables, but if the memsize is increased by another 50 interval (i.e. m = 150), then all the hit rates approach 100%.

In part2, the data follows the same principles as above, however, trace3 was supposed to have a hit rate of 0% in all three algorithms, but as you can see, my Clock algorithm does not. This result is not due to my implementation being incorrect, but because rather than use a counter that would act as a ‘clock hand’, I followed the algorithm described in the textbook, which simply picks a random frame and checks if it has its reference bit set to 1. The runtimes are the same, of course, I just thought this was more interesting than incrementing a ‘clock hand’. (I thought i’d just explain why there is discrepancy in my table for the Clock algorithm. Of course, if I did implement Clock with an iterator, then the hit rate would be 0%).