README for CSCC69 Assignment 2

Contents:

TABLES

With simpleloop.c, ran as "simpleloop":

Algorithm, Memory Size	Hit Rate	Hit Count	Miss Count	Overall Eviction Count	Clean Eviction Count	Dirty Eviction Count
RAND, 50	72.6772	8041	3023	2973	394	2579
RAND, 100	74.8644	8283	2781	2681	176	2505
RAND, 150	75.3886	8341	2723	2573	134	2439
RAND, 200	75.4519	8348	2716	2516	130	2386
FIFO, 50	73.1020	8088	2976	2926	325	2601
FIFO, 100	75.1085	8310	2754	2654	157	2497
FIFO, 150	75.4610	8349	2715	2565	129	2436
FIFO, 200	75.5333	8357	2707	2507	125	2382
LRU, 50	74.7921	8275	2789	2739	212	2527
LRU, 100	75.7683	8383	2681	2581	112	2469
LRU, 150	75.7683	8383	2681	2531	112	2419
LRU, 200	75.7683	8383	2681	2481	112	2369
CLOCK,50	74.7017	8265	2799	2749	220	2529
CLOCK,100	75.7411	8380	2684	2584	114	2470
CLOCK,150	75.7683	8383	2681	2532	112	2420
CLOCK,200	75.7683	8383	2681	2482	112	2370
OPT, 50	75.8496	8392	2672	2622	112	2510
OPT, 100	76.1659	8427	2637	2537	37	2500
OPT, 150	76.1659	8427	2637	2487	0	2487
OPT, 200	76.1659	8427	2637	2437	0	2437

With matmul.c, ran as "matmul 100":

Algorithm, Memory Size	Hit Rate	Hit Count	Miss Count	Overall Eviction Count	Clean Eviction Count	Dirty Eviction Count
RAND, 50	65.5656	1893785	994599	994549	955843	38706
RAND, 100	88.8092	2565150	323234	323134	315866	7268
RAND, 150	96.6621	2791972	96412	96262	94006	2256
RAND, 200	98.0382	2831720	56664	56464	54958	1506
FIFO, 50	60.9726	1761122	1127262	1127212	1083355	43857
FIFO, 100	62.4862	1804841	1083543	1063443	1061338	22105
FIFO, 150	98.8088	2853978	34406	34256	33060	1196
FIFO, 200	98.8269	2854499	33885	33685	32550	1135
LRU, 50	63.9509	1847149	1041235	1041185	1040210	975
LRU, 100	65.1553	1881935	1006449	1006349	1005388	961
LRU, 150	98.8615	2855501	32883	32733	31772	961
LRU, 200	98.8620	2855513	32871	32671	31710	961
CLOCK,50	63.9502	1847127	1041257	1041207	1040227	980
CLOCK,100	63.9583	1847360	1041024	1040924	1039961	963
CLOCK,150	98.8505	2855181	33203	33053	32089	964
CLOCK,200	98.8610	2855485	32899	32699	31737	962
OPT, 50	79.6614	2300928	587456	587406	586443	963
OPT, 100	96.7873	2795590	92794	92694	91732	962
OPT, 150	99.0787	2861773	26611	26461	25501	960
OPT, 200	99.3332	2869123	19261	19061	18101	960

With blocked.c, ran as "blocked 100 25":

Algorithm, Memory Size	Hit Rate	Hit Count	Miss Count	Overall Eviction Count	Clean Eviction Count	Dirty Eviction Count
RAND, 50	99.6527	2410200	8400	8350	5949	2401
RAND, 100	99.7818	2413323	5277	5177	3560	1617
RAND, 150	99.8163	2414156	4444	4294	2946	1348
RAND, 200	99.8406	2414745	3855	3655	2422	1233
FIFO, 50	99.7330	2412143	6457	6407	4286	2121
FIFO, 100	99.8201	2414250	4350	4250	2892	1358
FIFO, 150	99.8239	2414342	4258	4108	2804	1304
FIFO, 200	99.8682	2415413	3187	2987	2010	977
LRU, 50	99.7839	2413374	5226	5176	2945	2231
LRU, 100	99.8417	2414772	3828	3701	2767	961
LRU, 150	99.8419	2414777	3823	3673	2732	941
LRU, 200	99.8471	2414901	3699	3499	2558	941
CLOCK, 50	99.7825	2413340	5260	5210	2988	2222
CLOCK, 100	99.8328	2414555	4045	3945	2769	1176
CLOCK, 150	99.8353	2414616	3984	3834	2738	1096
CLOCK, 200	99.8665	2415371	3229	3029	2086	943
OPT, 50	99.8453	2414859	3741	3691	2729	962
OPT, 100	99.8746	2415568	3032	2932	1981	951
OPT, 150	99.8952	2416065	2535	2385	1426	959
OPT, 200	99.9055	2416315	2285	2085	1136	949

With fsh.c, a feeble shell program that mimics shell functionality.
Ran as "`fsh < file1`", where file1 is "`ls | head`"

Algorithm, Memory Size	Hit Rate	Hit Count	Miss Count	Overall Eviction Count	Clean Eviction Count	Dirty Eviction Count
RAND, 50	94.3371	9079	545	495	292	203
RAND, 100	97.5998	9393	231	131	22	109
RAND, 150	98.3583	9466	158	8	0	8
RAND, 200	98.3894	9469	155	0	0	0
FIFO, 50	95.2203	9164	460	410	210	200
FIFO, 100	97.8699	9419	205	105	1	104
FIFO, 150	98.3271	9463	161	11	0	11
FIFO, 200	98.3894	9469	155	0	0	0
LRU, 50	96.8516	9321	303	253	101	152
LRU, 100	98.2544	9456	168	68	1	67
LRU, 150	98.3894	9469	155	5	0	5
LRU, 200	98.3894	9469	155	0	0	0
CLOCK, 50	96.4048	9278	346	296	131	165
CLOCK, 100	98.1712	9448	176	76	2	74
CLOCK, 150	98.3479	9465	159	9	0	9
CLOCK, 200	98.3894	9469	155	0	0	0
OPT, 50	97.8387	9416	208	158	38	120
OPT, 100	98.3894	9469	155	55	0	55
OPT, 150	98.3894	9469	155	5	0	5
OPT, 200	98.3894	9469	155	0	0	0

Comparison of Algorithms:

After extensive testing, we now have evident data of the algorithms showing progress with every increase in memory size for each of the program traces. To varying degrees; the hit count increases, the miss count drops, and the overall eviction count drops by a notable amount. As for what algorithm performs the best, the data strongly supports what we have learned in the lectures. With consistent rankings to high hit rates and low eviction numbers, the general placing is as follows: OPT/Belady is in the lead, LRU is second, CLOCK is third, FIFO is fourth, and RAND is last. Considering that our OPT algorithm has the advantage of knowing the order of future pages, it should be no surprise that it has a significantly lower eviction count on most accounts, and subsequently, a higher hit rate. It is also rather obvious that RAND has a high eviction count and a low hit rate due to its haphazard method in choosing a victim to evict. The only odd outlier to this is the “matmul 100” case on the memory sizes of 50 and 100, which have the RAND algorithm above CLOCK, FIFO, and even LRU. While there are obvious improvements happening in parallel with the memory size, there is also a plateau in how much increasing it can prove to be beneficial. Our custom “fsh” program is where this is most demonstrable. In it, you'll every see every algorithm stop increasing its hit rate once it reaches 98.3894, and likewise, you'll see it remain at 0 for the eviction counts. Clearly, for smaller trace files and/or higher memory, the discrepancies between the algorithms is less pronounced.

LRU Algorithm:

In running the traces through the LRU algorithm, it would seem that increasing the amount of memory results in more efficient numbers across the board. This was expected, considering that our LRU implementation is only dependent on time stamps (as opposed to a reshuffling stack), so increasing the frame amount would just allow for more pages to be held and result in more hits. This is most evident in the difference between the 100 and 150 memory sizes in the “matmul 100” table; where the hit rate drastically improves from 65.1553 to 98.8615. Additionally, the huge decrease of eviction counts (1006349 to 32733 in the mentioned “matmul 100” case) can also be explained because the LRU algorithm now has less need for pages to be evicted. LRU also seems to peak rather early in certain cases, such as its hit rate remaining at 75.7683 on memory size 100 and onward for the “simpleloop” case. This can be explained by the diminishing returns in memory size for all the algorithms noted earlier, and since LRU is much closer to Belady's algorithm, there's a lot less improvement to work with for a higher memory pool.