Blocked	Hit Rate	Hit Count	Miss Count	Overall eviction count	clean eviction count	dirty eviction count
50 clock	99.7818	2412876	5276	5226	1466	3760
50 fifo	99.73	2411622	6530	6480	2135	4345
50 lru	99.7839	2412926	5226	5176	1439	3737
50 opt	99.8465	2414439	3713	3663	1325	2338
50 rand	99.6516	2409727	8425	8374	3071	5304
100 clock	99.8338	2414232	4020	3920	1335	2585
100 fifo	99.8204	2413810	4342	4242	1399	2843
100 lru	99.8433	2414363	3789	5017	1328	2361
100 opt	99.8754	2415139	3013	2912	1045	1868
100 rand	99.7834	2412915	5237	5137	1833	3304
150 clock	99.8368	2414206	3946	4631	1327	3304
150 fifo	99.8251	2413923	4229	4079	1367	2712
150 lru	99.844	2414380	3772	3622	1319	2303
150 opt	99.8953	2415621	2531	2381	833	1548
150 rand	99.8188	2413771	4381	4231	1544	2687
200 clock	99.8671	2414939	3213	3013	1081	1932
200 fifo	99.8685	2414973	3179	2979	1003	1976
200 lru	99.847	2414453	3699	3499	1281	2218
200 opt	99.9057	2415871	2281	2081	663	1418
200 rand	99.8402	2414287	3865	3665	1323	2342
matmul	Hit Rate	Hit Count	Miss Count	Overall eviction count	clean eviction count	dirty eviction count
50 clock	63.9455	1846778	1041270	1041220	520117	521103
50 fifo	60.9671	1760760	1127288	1127238	541691	585547
50 lru	63.9468	1846814	1041234	1041184	520104	521080
50 opt	79.6589	2300588	587460	587410	293416	293994
50 rand	65.5334	1892637	995411	995361	478159	517202
100 clock	63.9511	1846938	1041110	1041010	520031	520979

100 fifo	62.4816	1804499	1083549	1083449	530672	552777
100 lru	65.1509	1881589	1006459	1006359	502792	503567
100 opt	96.7868	2795248	92800	92700	46011	46689
100 rand	88.7743	2563843	324205	324105	158540	165565
150 clock	98.8499	2854832	33216	33066	16134	16932
150 fifo	98.8085	2853636	34412	34262	16665	17597
150 lru	98.8612	2855158	32890	32740	16018	16722
150 opt	99.0784	2861431	26617	26467	12929	13538
150 rand	96.6636	2791690	96358	96208	47113	49095
200 clock	98.8606	2855141	32907	32707	15988	16719
200 fifo	98.8265	2854157	33891	33691	16250	17441
200 lru	98.8616	2855170	32878	32678	15985	16693
200 opt	99.3329	2868781	19267	19067	9240	9827
200 rand	98.045	2831586	56462	56262	27496	28766
plus	Hit Rate	Hit Count	Miss Count	Overall eviction count	clean eviction count	dirty eviction count
50 clock	97.0578	5839	177	127	25	102
50 fifo	96.26	5791	225	175	43	132
50 lru	97.3737	5858	158	108	19	89
50 opt	97.9887	5895	121	71	3	68
50 rand	96.1769	5786	230	180	49	131
100 clock	98.0386	5898	118	18	0	18
100 fifo	97.889	5889	127	27	0	27
100 lru	98.0884	5901	115	15	0	15
100 opt	98.1051	5902	114	14	0	14
100 rand	97.9887	5895	121	21	1	20
150 clock	98.1051	5902	114	0	0	0
150 fifo	98.1051	5902	114	0	0	0
150 lru	98.1051	5902	114	0	0	0

150 opt	98.1051	5902	114	0	0	0
150 rand	98.1051	5902	114	0	0	0
200 clock	98.1051	5902	114	0	0	0
200 fifo	98.1051	5902	114	0	0	0
200 lru	98.1051	5902	114	0	0	0
200 opt	98.1051	5902	114	0	0	0
200 rand	98.1051	5902	114	0	0	0
simpleloop	Hit Rate	Hit Count	Miss Count	Overall eviction count	clean eviction count	dirty eviction count
50 clock	72.9151	7554	2806	2756	71	2685
50 fifo	71.168	7373	2987	2937	131	2806
50 lru	73.0502	7568	2792	2742	68	2674
50 opt	74.2181	7689	2671	2621	20	2601
50 rand	71.0232	7358	3002	2952	147	2805
100 clock	74.0347	7670	2690	2590	5	2585
100 fifo	73.3591	7600	2760	2660	33	2627
100 lru	74.0637	7673	2687	2587	2	2585
100 opt	74.4788	7716	2644	2544	0	2544
100 rand	73.2046	7584	2776	2676	41	2635
150 clock	74.0541	7672	2688	2538	0	2538
150 fifo	73.7548	7641	2719	2569	8	2561
150 lru	74.083	7675	2685	2535	0	2535
150 opt	74.4788	7716	2644	2494	0	2494
150 rand	73.8031	7646	2714	2564	9	2555
200 clock	74.0734	7674	2686	2486	0	2486
200 fifo	73.832	7649	2711	2511	6	2505
200 lru	74.083	7675	2685	2485	0	2485
200 opt	74.4788	7716	2644	2444	0	2444
200 rand	73.7548	7641	2719	2519	8	2511

1. 4th program

For our fourth program we wanted to see a simulation for a program that continuously assessed the same pages multiple times across different memsizes. We used this because the behaviour was different than the other programs which ranged from the simple loop and storing matrices for matrix multiplication. Our results adhered to the same pattern we found below, that fifo < clock< lru <rand.

2. Results

Opt unsurprisingly always had the best hit rate percentage across all programs and memsizes so we will ignore it for the rest of this discussion. In blocked, lru performed the best except until size 200 where it is outmatched by both fifo and clock. At each size, clock performed better than fifo. In Matmul, LRU performed better at size 150 onwards with rand having the best non opt performance previously. Clock always outperformed fifo and in general clock and lru performed similarly. In our plus program, we saw the consistent pattern of RAND < FIFO < CLOCK < LRU <OPT across all sizes. Finally in simple loop, we saw two generally patterns:
RAND < FIFO < LRU< CLOCK < OPT for sizes of 50,100
RAND < FIFO < CLOCK < LRU < OPT for sizes of 150, 200
Overall ignoring rand, our pattern for performance is:
FIFO < CLOCK and LRU (had similar performance) < OPT.

3. LRU analysis

In general, we observed that as memsize increased the hit rate percentage also increased, eventually becoming very close to the optimal hit rate.

This can be explained in the way LRU is implemented. One of the biggest benefits of using the LRU algorithm is that it is based on spatial locality. A rule which states that any page that was referenced recently is likely to be referenced again soon. This is precisely why we evict the page with the lowest counter/timestamp attached to it (see our implementation). Logically this page was used less recently than all other pages (it was used farthest back when looking in the past) so we can assume that it won't be referenced anytime in the near future.

So as memsize is increased, there are more frames in our physical memory (coremap). This means our coremap can store more frames before having to try to evict any. This is ideal for how LRU is designed. Since we store the variables for 'how recently used' within that frame, having more frames in memory when trying to evict means we have a better approximation of which was the one least recently used.

For example having 100 frames, and a core map that can hold 10 frames vs 50. With 10 we can only approximate which 10 out of the total number was the least recently used, but with a larger sample size of say 50, we may be able to find a frame that was used even less recently or closer to the actual least recently used when trying to evict. Simply put more memsize means more frames to compute the least recently used (compare on eviction) meaning a more accurate deduction of which frame to evict.

When looking at how it compared to the other algorithms (not including opt), generally LRU was the best performing across all memsizes save for clock which performed similarly. We can see this in Blocked, where at memsize 200, it fell behind fifo and clock and for simpleloop where at memsize 100, fell behind fifo. For all traces, LRU performed very closely to the opt performance, except in matmul for low sizes.

Finally, for matmul, we see that lru performs very poorly at memsizes of both 50 and 100, then sees a massive jump in accuracy at memsize 150 to be close to the opt hit rate. We can deduce that for lru to evict the ‘least recently used’ accurately it needs a memsize greater than 100. Finally we can also argue that 150 is the best memsize for the algorithm as when observing memsize 200, we actually see a slight reduction in hit percentage, meaning its closeness to the optimal solution is reduced.