tr-simpleloop.ref
swapfile size = 2644

LOAD: 61
STORE: 2514
MODIFY: 1
TOTAL INSTR: 2644

memsize: 50

	HitRate	HitCount	MissCount	CleanEvict	DirtyEvict	TotalEvict
rand	73.0625	8183	3017	156	2811	2967
fifo	73.4107	8222	2978	127	2801	2928
lru	75.1161	8413	2787	68	2669	2737
clock	75.0000	8400	2800	68	2682	2750
opt	76.0714	8520	2680	25	2605	2630

memsize: 100

	HitRate	HitCount	MissCount	CleanEvict	DirtyEvict	TotalEvict
rand	75.0804	8409	2791	49	2642	2691
fifo	75.3304	8437	2763	32	2631	2663
lru	75.9732	8509	2691	2	2589	2591
clock	75.9554	8507	2693	3	2590	2593
opt	76.4286	8560	2640	0	2540	2540

memsize: 150

	HitRate	HitCount	MissCount	CleanEvict	DirtyEvict	TotalEvict
rand	75.6875	8477	2723	13	2560	2573
fifo	75.6964	8478	2722	8	2564	2572
lru	76.0000	8512	2688	0	2538	2538
clock	75.9911	8511	2689	0	2539	2539
opt	76.4286	8560	2640	0	2490	2490

memsize: 200

	HitRate	HitCount	MissCount	CleanEvict	DirtyEvict	TotalEvict
rand	75.7054	8479	2721	12	2509	2521
fifo	75.7679	8486	2714	6	2508	2514
lru	76.0000	8512	2688	0	2488	2488
clock	75.9911	8511	2689	0	2489	2489
opt	76.4286	8560	2640	0	2440	2440

tr_matmul.ref
swapfile size = 1097

LOAD: 66
STORE: 953
MODIFY: 1
TOTAL INSTR: 1097

memsize: 50

	HitRate	HitCount	MissCount	CleanEvict	DirtyEvict	TotalEvict
rand	65.5643	1894000	994768	478034	516684	994718
fifo	60.9777	1761504	1127264	541683	585531	1127214
lru	63.9562	1847547	1041221	520108	521063	1041171
clock	63.9560	1847540	1041228	520107	521071	1041178
opt	79.6639	2301306	587462	293422	293990	587412

memsize: 100

	HitRate	HitCount	MissCount	CleanEvict	DirtyEvict	TotalEvict
rand	88.7958	2565105	323663	158195	165368	323563
fifo	62.4910	1805219	1083549	530669	552780	1083449
lru	65.1597	1882313	1006455	502789	503566	1006355
clock	65.3211	1886974	1001794	500464	501230	1001694
opt	96.7877	2795971	92797	46011	46686	92697

memsize: 150

	HitRate	HitCount	MissCount	CleanEvict	DirtyEvict	TotalEvict
rand	96.6553	2792147	96621	47242	49229	96471
fifo	98.8089	2854359	34409	16664	17595	34259
lru	98.8616	2855882	32886	16017	16719	32736
clock	98.6047	2848461	40307	19757	19757	39514
opt	99.0787	2862154	26614	12927	13537	26464

memsize: 200

	HitRate	HitCount	MissCount	CleanEvict	DirtyEvict	TotalEvict
rand	98.0341	2831979	56789	27648	28941	56589
fifo	98.8269	2854880	33888	16248	17440	33688
lru	98.8620	2855894	32874	15983	16691	32674
clock	98.8615	2855880	32888	15994	16694	32688
opt	99.3331	2869504	19264	9238	9826	19064

tr_blocked.ref
swapfile size = 1099

LOAD: 67
STORE: 953
MODIFY: 1
TOTAL INSTR: 1099

memsize: 50

	HitRate	HitCount	MissCount	CleanEvict	DirtyEvict	TotalEvict
rand	99.6569	2410549	8299	3019	5230	8249
fifo	99.7343	2412421	6427	2084	4293	6377
lru	99.7878	2413715	5133	1408	3675	5083
clock	99.7629	2413112	5736	1662	4024	5686
opt	99.8471	2415149	3699	1325	2324	3649

memsize: 100

	HitRate	HitCount	MissCount	CleanEvict	DirtyEvict	TotalEvict
rand	99.7848	2413642	5206	1831	3275	5106
fifo	99.8219	2414541	4307	1380	2827	4207
lru	99.8435	2415063	3785	1325	2360	3685
clock	99.8302	2414742	4106	1326	2680	4006
opt	99.8761	2415850	2998	1041	1857	2898

memsize: 150

	HitRate	HitCount	MissCount	CleanEvict	DirtyEvict	TotalEvict
rand	99.8195	2414481	4367	1528	2689	4217
fifo	99.8260	2414639	4209	1356	2703	4059
lru	99.8442	2415079	3769	1316	2303	3619
clock	99.8437	2415067	3781	1323	2308	3631
opt	99.8957	2416325	2523	823	1550	2373

memsize: 200

	HitRate	HitCount	MissCount	CleanEvict	DirtyEvict	TotalEvict
rand	99.8404	2414988	3860	1321	2339	3660
fifo	99.8692	2415685	3163	994	1969	2963
lru	99.8472	2415152	3696	1279	2217	3496
clock	99.8676	2415646	3202	1062	1940	3002
opt	99.9060	2416575	2273	655	1418	2073

tr-heaploop mine
swapfile size = 268

LOAD: 60
STORE: 139
MODIFY: 1
TOTAL INSTR: 268

memsize: 50

	HitRate	HitCount	MissCount	CleanEvict	DirtyEvict	TotalEvict
rand	92.9516	7451	565	135	38	173
fifo	93.5504	7499	517	110	357	467
lru	94.8728	7605	411	68	293	361
clock	94.6856	7590	426	67	309	376
opt	96.2076	7712	304	25	229	254

memsize: 100

	HitRate	HitCount	MissCount	CleanEvict	DirtyEvict	TotalEvict
rand	95.6961	7671	345	28	217	245
fifo	95.7460	7675	341	20	221	241
lru	96.0704	7701	315	2	213	215
clock	96.0329	7698	318	3	215	218
opt	96.7066	7752	264	0	164	164

memsize: 150

	HitRate	HitCount	MissCount	CleanEvict	DirtyEvict	TotalEvict
rand	96.3448	7723	293	3	140	143
fifo	96.0704	7701	315	0	165	165
lru	96.2201	7713	303	0	153	153
clock	96.0953	7703	313	0	163	163
opt	96.7066	7752	264	0	114	114

memsize: 200

	HitRate	HitCount	MissCount	CleanEvict	DirtyEvict	TotalEvict
rand	96.5694	7741	275	0	75	75
fifo	96.2201	7713	303	0	103	103
lru	96.5444	7739	277	0	77	77
clock	96.2450	7715	301	0	101	101
opt	96.7066	7752	264	0	64	64

Comparison:

Opt has the highest hit rate for all the algorithms. Followed by lru then clock then fifo. The random hit rate varies randomly. The reason opt has the highest hit rate is because it has the highest number of hits all the time. You would expect this because opt tries to only evict dirty frames that will not be used again or is the furthest from use. Lru follows in second, also understandable because it tries it tries to evict the frames that haven't been used the longest. With lru the hit rate would probably increase as the memory size grows closer to the swap file size. Clock and fifo come in at third and fourth. You will notice that clock and fifo hit rates get close to each other as the memory sizes increases. My analysis is that since the memory size is large, only a few frames will have their reference changed back to 1 and then most of the first introduced frames will be the first to leave.

Naturally as the memory size increases the number of evictions will go down. But you will notice also that the number of clean evictions go down as well. This is because if a frame has not been modified at all and is evicted then that is a clean eviction. I assume that when the memory size is large, evictions and more likely to be dirty because pages are bound to be modified before the memory is full and an eviction is needed.

LRU description:

LRU gets a much better hit rate as the memory size increases. Unlike opt, lru is contingent on the size of the memory. You will notice that lru always looks for the frame in memory that has hasn't been used the longest. But if you have a small memory size the hit rate is only as high as the average number of instructions till any process is used again. Especially if the swap file size is really large then the lru may evict frames that will be used fairly soon compared to the size of large swap file size. But as the size of the memory increases then lru becomes more accurate. Because the lru estimate of how long a frame hasn't been used becomes more accurate.