Documentação Trabalho Pratico 3

Estevão Moraes de Andrade
Gabriel Alves Fernandes

Introdução

O problema proposto é desenvolver um simulador de memória virtual . Para tanto foram implementadas soluções que focam em testar e simular o funcionamento dos algoritmos de reposição sem focar a eficiencia quanto a implementação , mas buscando reduzir a complexidade do codigo de tal maneira que ele execute com um tempo reduzido.

Para isso foi criado uma estrutura “page” que reflete as caracteristicas de uma pagina que será reposta. Ela é a principal variavel do programa porque define o endereço , o ultimo acesso e o tipo de acesso (leitura / escrita) .

Para simular a tabela de paginas é criado um vetor que é alocado dinamicamente para o tamanho de paginas possíveis. Esse tamanho é calculado , baseado no tamanho da pagina , logo considerando 32bits , uma pagina que possua x bits , o tamanho da tabela de paginas é de 32bits – x . Nessa estrutura é armazenado o indice da pagina na memoria fisica . E o indice desse vetor identifica a pagina , essa escolha de implementação foi com proposito de reduzir a complexidade do acesso a pagina , para O(1) , tornando mais rapido o acesso aos dados na memória .

Como dito antes , a memoria fisica é utilizada para armazenar as paginas , para isso , é criado um vetor do tipo “page” que recebe as paginas a serem colocadas na memória . Essa variavel é uma simulação da memória fisica , para isso , o indice do vetor de memoria é armazenado no vetor de tabelas de paginas , onde o indice é a pagina que será carregada .

Para tratar do tipo de acesso e contabilizar page faults e paginas escritas . Existem duas variaveis globais chamadas page_faults e paginas_escritas. Page_faults é contabilizado a cada acesso na memória , caso a pagina não esteja alocada é incrementado essa variavel . Já paginas_escritas é incrementada a cada momento que uma pagina foi alterada na memória . Nesse caso , para considerar a alteração de uma pagina , toda vez que uma pagina com acesso “W” é retirada da memória fisica , a variavel paginas_escritas é incrementada , assim oferecendo a estatistica no final do programa.

Quanto a estrutura do arquivo , para modularização do codigo , os arquivos são separados por sua estrutura ou funções basicas . Assim , existe a seguinte treefile

Root

- main.c : Arquivo contendo conteudo de execução do codigo
- page.h : Arquivo com a estrutura e funções relacionadas a paginas
- funcoes.c : Arquivo com as funções principais do codigo , inclusive metodos de reposição
- memoria.h : Arquivo contendo a variavel global MEMORIA e funções que trabalhem sobre ela.
- tabela.h : Arquivo contendo a variavel global TABELA_PAGINAS e as funções que trabalhem sobre ela
- globais.h : As variaveis globais de estatistica

Metodos de Reposição

Os metodos de reposição são baseados nos dois vetores já descritos anteriormente . O vetor de memória possui as estruturas page e para avaliar qual pagina deve ser retirado é utilizada um campo dentro da pagina . É um valor inteiro que é contabilizado a cada acesso ao arquivo de log, dessa maneira cada pagina recebe um tempo de acesso e assim é possível utilizar o metodo FIFO e LRU sem necessidade de implementar uma fila ou algo do tipo.

FIFO

O metodo retira o primeiro item inserido , independente de hits postériores ou não , para tanto , a cada acesso a uma page , ela é verificada na tabela de paginas se está na memória e caso esteja a complexidade para buscar essa pagina é O(1) . Dessa maneira ocorre um hit ou no caso de page fault é feita uma pesquisa na memória e avaliado a pagina com tempo menor.

LRU

Esse metodo retira a pagina mais antiga , a diferença vital entre o FIFO é que a cada hit a pagina na memória tem o tempo atualizado , assim garantindo que o ultimo acessado realmente seja retirado .

RANDOM

O metodo simplesmente seleciona um valor aleatório e retira da memória o indice que foi selecionado.

Resultados

Os resultados para cada metodo de reposição serão mostrados em uma tabela com duas linhas , uma para as page faults e um para as paginas escritas . Dessa maneira serão 2 graficos para cada. Considerando que um grafico deve variar o tamanho da pagina na seguinte sequência (2,4,8,16,32,64) e o tamanho da memória fisica na seguinte sequência (128 , 256, 512 , 1024, 2048 , 4096 , 8192 , 16384).

Para melhor identificar essa analise o grafico pode ser lido de duas maneiras , a curva da linha representa para cada tamanho de memoria a variação do tamanho da pagina . Porém caso a leitura seja interessada em visualizar o tamanho fixo de uma pagina relacionado ao crescimento do tamanho da memória , basta analisar os pontos naquele tamanho de pagina e verificar a altura de cada ponto.

Analisando de uma maneira geral os graficos , foi possível identificar um padrão comum para todos os metodos de reposição , a medida que o tamanho de pagina crescia a quantidade de page faults e paginas escritas cresciam com eles . Porém esse crescimento era variavel a medida que a memória também crescia .

Considerando assim para o caso de uma memória de 128 kb a curva com maior crescimento.

Para os padrões de reposição , abaixo trataremos das especificidades apresentadas nos seus graficos em comparação aos demais.

FIFO

Como já esperado esse metodo de reposição foi o que teve os piores resultados , apesar do metodo random configurar um padrão aleatório , até o mesmo possuiu resultados mais agradaveis do que o FIFO .

Para o fifo a inclinação da curva foi bem maior do que os demais se avaliando cada alteração de tamanho de memória . E também aquele que possuiu a maior diferença de page faults quando haviam a alteração do tamanho da memoria fixando um tamanho de pagina . Especialmente essa curva aumentava bastante quando a memoria era alterada e o tamanho da pagina permanecia em 32KB.

Um peculiaridade aconteceu , a conhecida anomalia de belamy foi encontrada em uma situação . Para o arquivo de matriz.log quando a tamanho da pagina foi fixada em 32kb , o tamanho de memoria 128kb teve menos page faults do que uma memoria de tamanho 256 kB.

Outra fato observado é que para o programa compressor ao atingir uma memória de tamanho 16mb , 8mb e 4mb a medida que o tamanho da pagina é aumentado , ao inves de aumentar o número de pages faults , ele diminui , contrariando a logica de que menos paginas disponiveis , deveriamos haver mais page faults.

LRU

Também como esperado , o metodo de reposição LRU foi o mais eficiente , o programa que apresentou maior quantidade de page faults foi o matriz.log que possuiu por volta de 370 mil page fauls . Se comparado o mesmo grafico aos demais metodos de reposição foi evidentemente o mais eficiente , considerando que os demais tiveram valores na cada dos 470 mil . Para isso esse valores são considerados para tamanho de pagina 64kb e tamanho de memoria 128kb . Porém é possível estender esse resultado para os demais tamanhos de memória e paginas pois , como mostra nos graficos os crescimentos seguem um padrão comum , alterando somente sua taxa de variação baseado no tamanho da memoria . Ou seja memorias maiores possuem uma variação menor da curva de crescimento.

Outro fato interessante é que em alguns casos as paginas escritas retiradas

da memoria foram nulas . Ou seja , aquelas que foram escritas inicialmente não sairam da fila de memória.

Acompanhando um fato que ocorreu no FIFO , porem agora em outro programa, no simulador.log , para um tamanho de memória 16Mb ao aumentar o tamanho das paginas os page faults cairam , contrariando a expectativa. Esse fato ainda se repetiu no programa compressor e matriz , onde a memoria estando em 16Mb os valores de page fault cairam a medida que foir aumentado o tamanho das paginas.

RANDOM

Seguindo o padrão de crescimento dos outros metodos de reposição , apesar do metodo aleatorio não possuir uma expectativa anterior , ele seguiu um padrão estavel , se compararmos o grafico dos 4 programas é possível verificar que os maiores crescimentos de page faults foram realmente de tamanhos de memória entre 128Kb e 512Kb , a medida que o tamanho das paginas iam crescendo.

Analisando esse fato é possível perceber que a medida que é acrescida a memoria , independente do tamanho da pagina a diferença entre os page faults é bem pequena . Porém , não um dado levantado , mas ao longo dos experimentos foi possível perceber que para memorias muito grandes o tempo de execução do programa realmente era maior , partindo desse ponto , é possível encontrar um ponto de equilibro entre o tempo de execução de um programa e a quantidade de memoria .

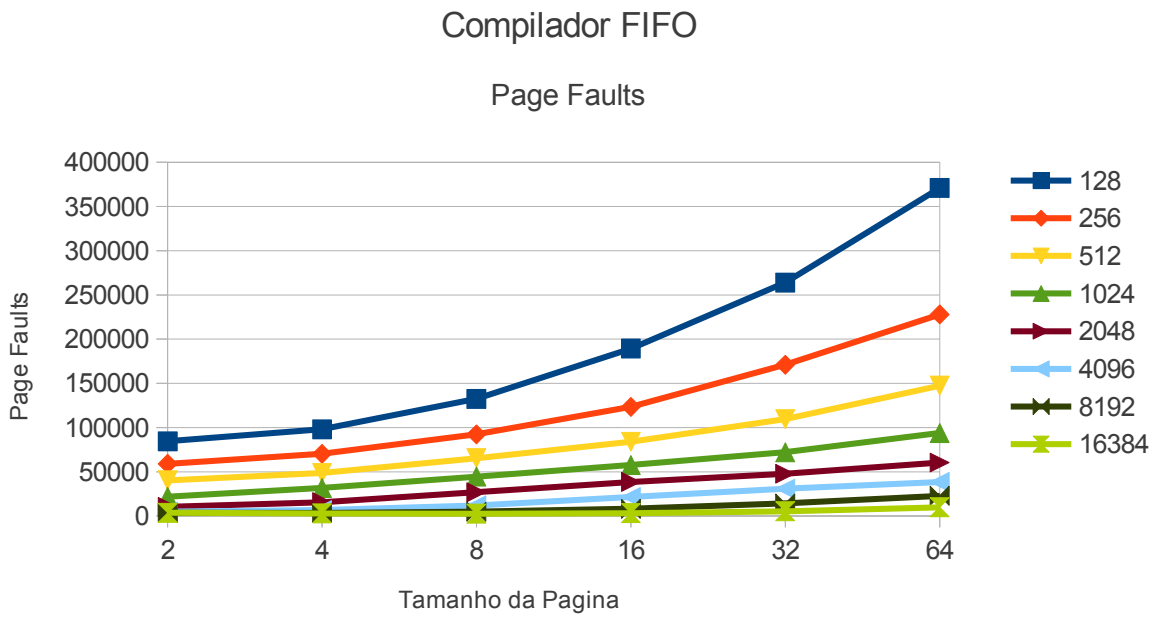
Conclusão geral

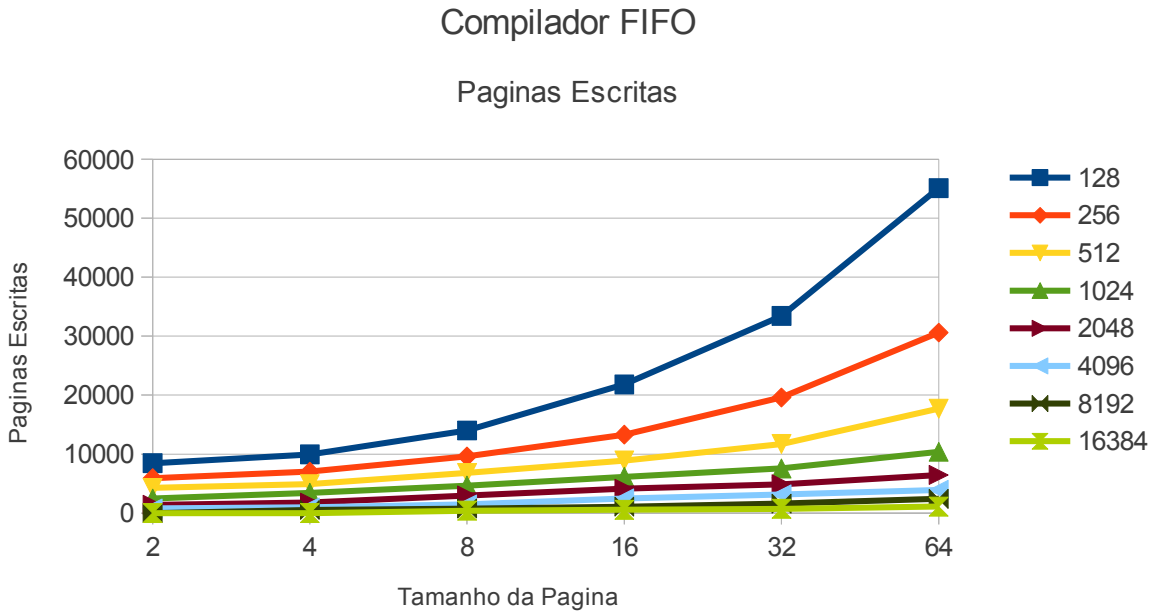
Após a analise dos graficos abaixo e da serie de testes foi possível perceber que o metodo de reposição de paginas LRU é o mais eficiente , não apenas por questões de page faults , mas também pela retirada de paginas escritas . Se considerarmos que em alguns casos essas paginas escritas não foram retiradas da memória podemos considerar que há um problema no sentido de ter um disco com dados desatualizados , porém , para vias de testes , a não seleção dessas paginas demonstra que o programa le o maior número de informações diferentes, não que isso seja onus do simulador , mas sim do metodo de reposição .

A relação tempo x tamanho de memória também foi percebido . Para programas que executaram em tamanhos de memória muito grandes como 16Mb o tempo de execução foi bem superior se comparado a execução de uma memória de 512 kb , entretando , em alguns casos , como no LRU e fixando o tamanho da pagina , foi possível perceber que a quantidade de pages faults era bem pequena , mostrando que o tamanho da memória não influenciava tanto assim no resultado . Dado esses fatos é possível presumir que para os arquivos de logs testado a melhor configuração , seria roda-los em uma memória de 2MB com tamanho de pagina de 8kb. Pois em todos os graficos foi possível perceber que a diferença de page faults para memorias maiores é bem próximo e para os tamanhos de pagina a mesma coisa . Consideramos assim , a configuração mais eficaz.

Abaixo seguem os graficos que dão base a essa analise , lembrando que os valores estão escritos em uma tabela abaixo do grafico . Essa tabela retem os valores exatos de page faults e paginas escritas . As analises feitas nesse trabalho são baseados no comportamento das linhas nesses graficos e dos resultados descritos nessa tabela . Como o grafico não reflete bem o detalhe dos valores , por esses serem muito grandes , a tabela serve de auxilio para determina eventos de valores menores , mas ainda assim importantes.

FIFO
compilador.log

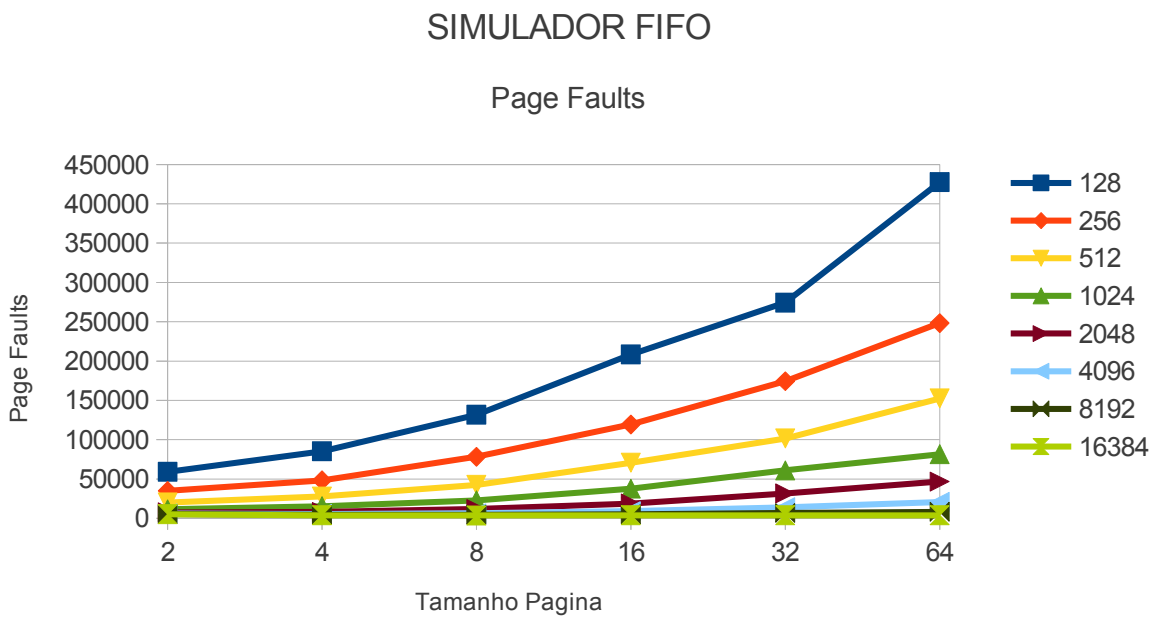


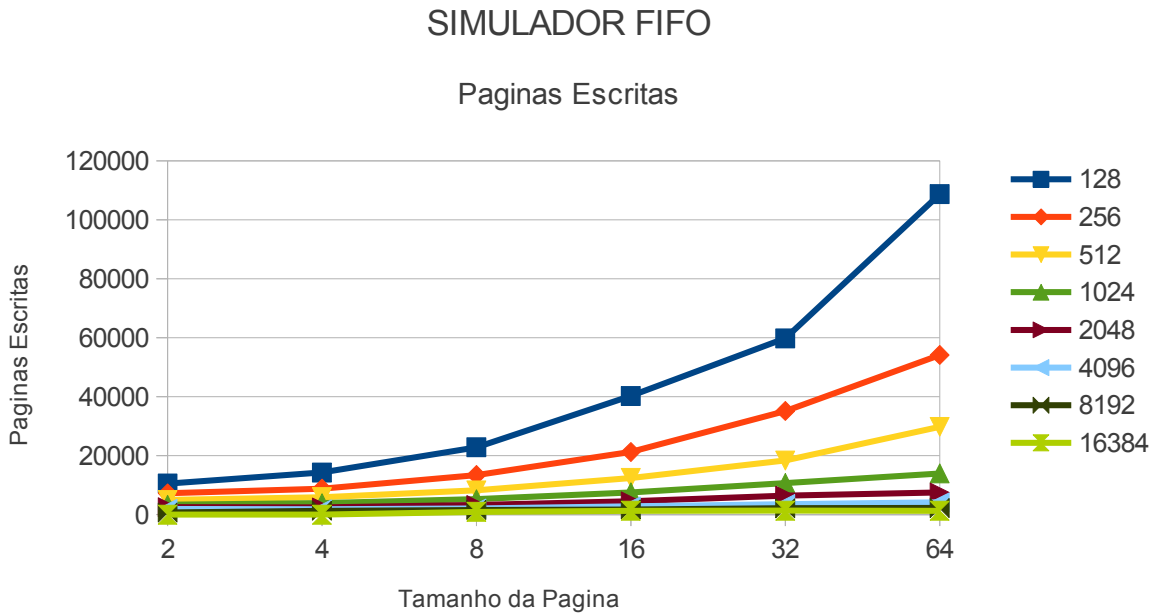


FIFO	PAGE FAUL							
	128	256	512	1024	2048	4096	8192	16384
2	84419	59035	40341	21639	10190	4849	3621	3621
4	98067	70315	48526	31698	15609	6673	3432	2852
8	132386	92279	65270	44465	26928	11951	4825	2494
16	189322	123455	83950	57600	38430	21527	8198	3201
32	263907	170963	109458	72140	47785	30752	14172	5223
64	370856	227915	147332	93778	60369	38496	22804	9859

FIFO	PAGINAS ESCRITAS							
	128	256	512	1024	2048	4096	8192	16384
2	8446	5912	4261	2448	1388	779	0	0
4	9935	7018	4911	3425	1776	959	517	0
8	13997	9596	6797	4649	2976	1464	757	389
16	21806	13291	8905	6144	4126	2465	1070	555
32	33417	19598	11706	7566	4876	3142	1579	718
64	55109	30603	17686	10386	6451	3927	2400	1100

simulador.log

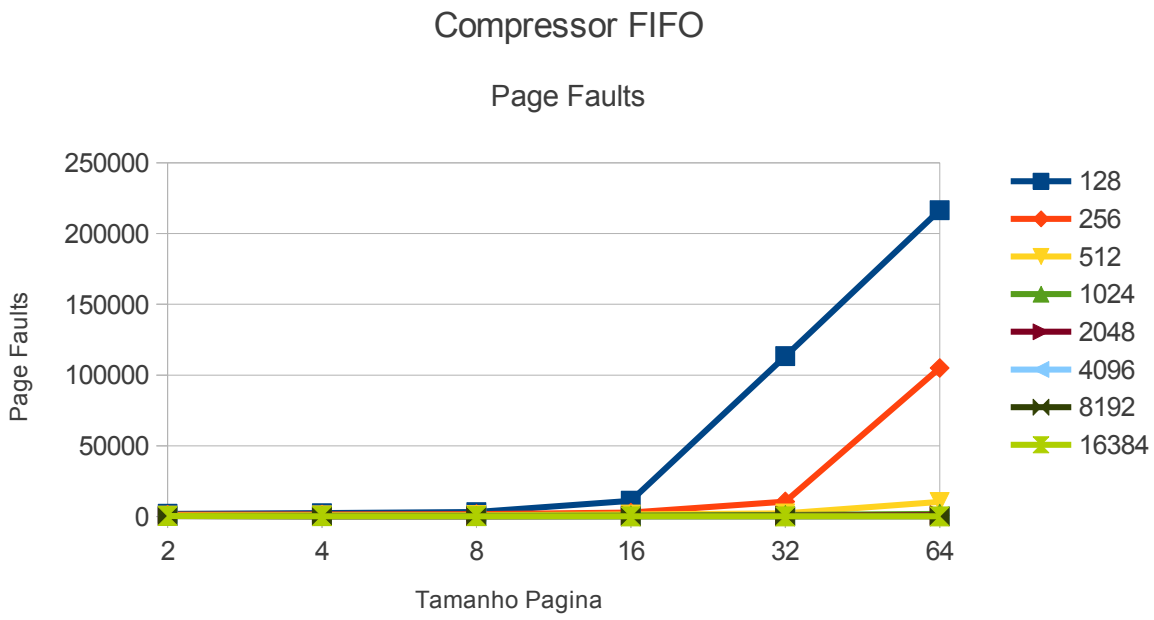


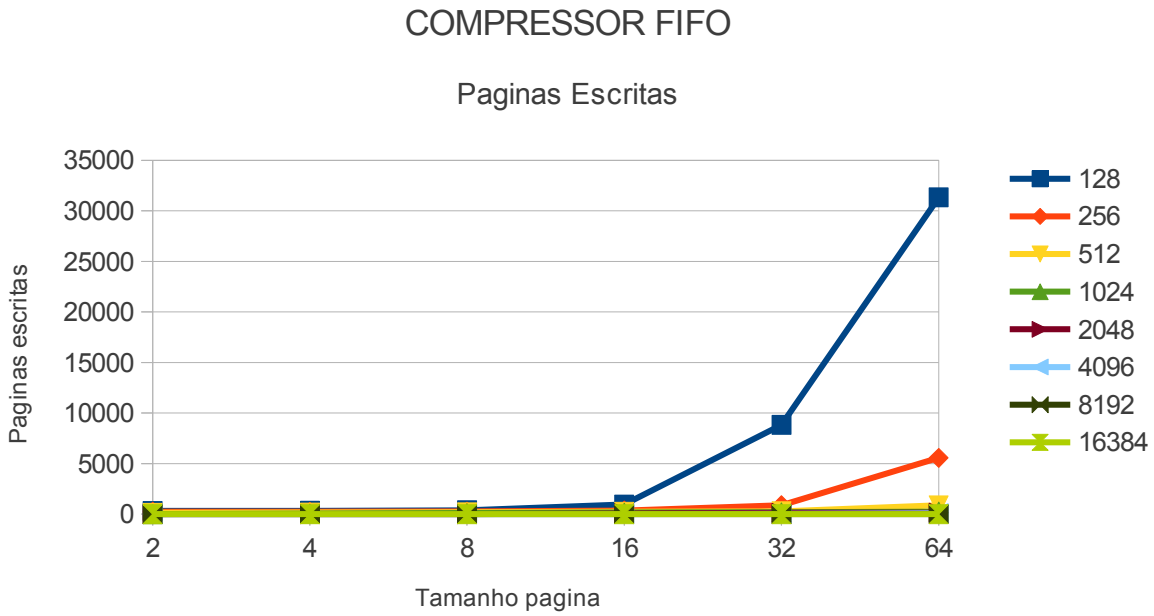


FIFO	PAGE FAUL							
	128	256	512	1024	2048	4096	8192	16384
2	59155	34943	20163	11449	7663	6168	5333	5144
4	85283	48301	27778	15440	8089	5492	4314	3890
8	131827	78338	42421	22816	11884	6293	4528	3669
16	208367	119307	70724	37606	18800	9188	5108	3751
32	274439	174434	101483	60922	31700	14066	6783	3837
64	427573	248262	152352	81506	46770	20809	8367	3934

FIFO	PAGINAS ESCRITAS							
	128	256	512	1024	2048	4096	8192	16384
2	10541	7236	4955	3373	2539	1972	740	0
4	14301	8831	5936	3886	2444	1832	1314	0
8	22870	13446	8218	5239	3276	2095	1571	952
16	40255	21270	12439	7570	4532	2797	1792	1319
32	59832	35136	18368	10736	6453	3579	2158	1370
64	108697	54149	29779	13968	7517	4152	2267	1331

compressor.log

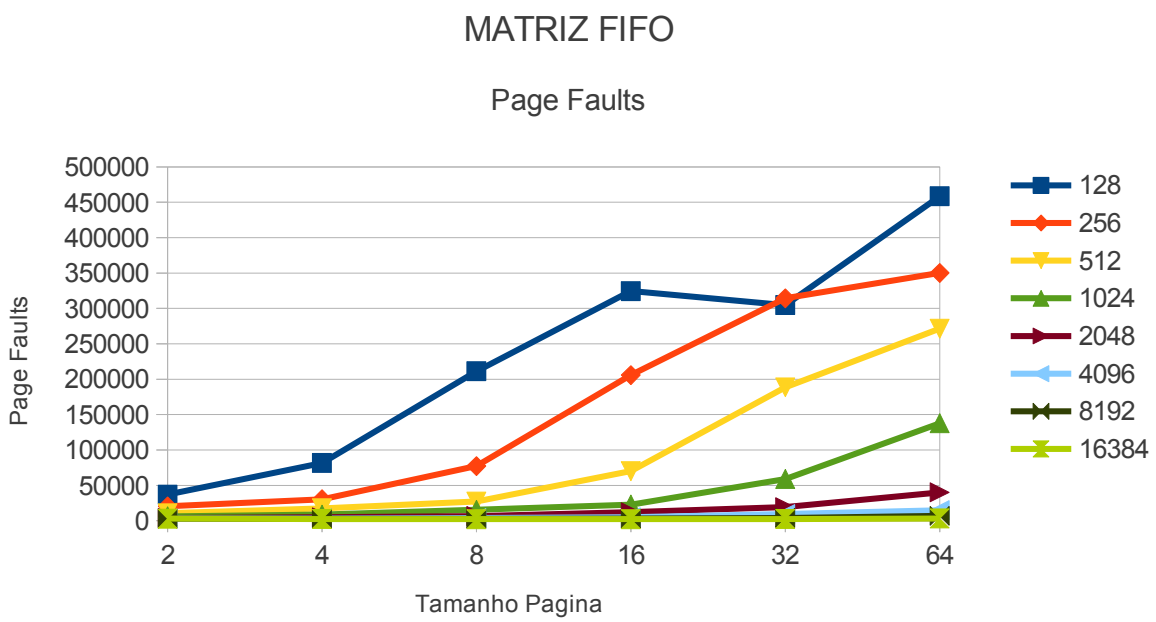


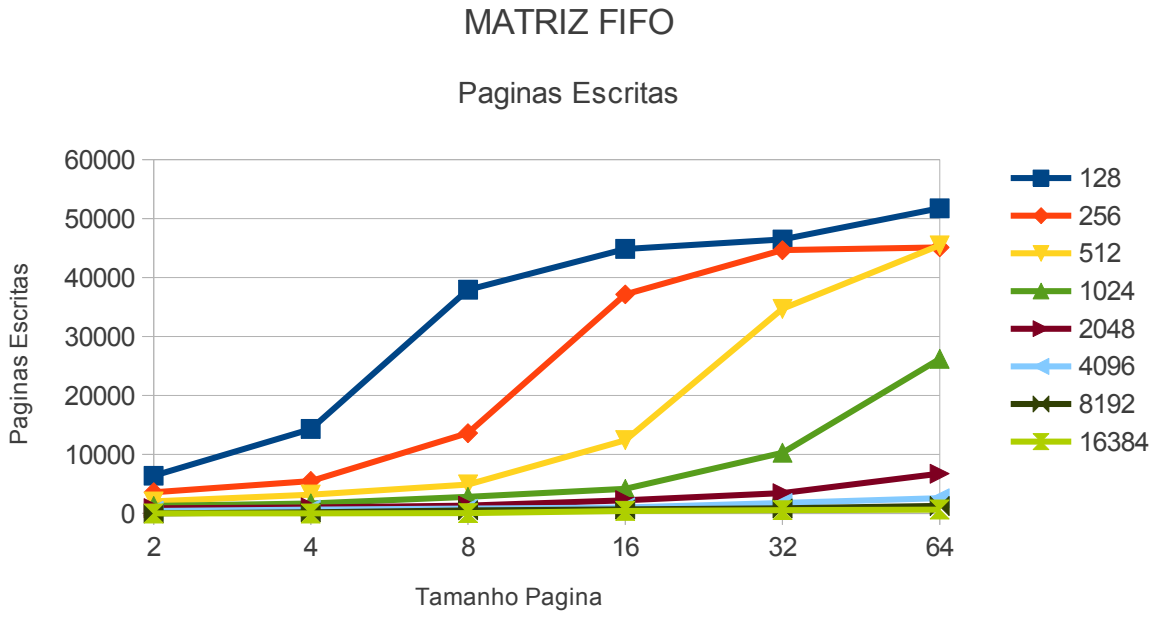


FIFO	PAGE FAUL							
	128	256	512	1024	2048	4096	8192	16384
2	1928	1249	772	419	419	419	419	419
4	2497	1467	891	511	317	317	317	317
8	3235	1804	1030	570	255	255	255	255
16	11214	2862	1495	850	430	209	209	209
32	113347	10640	2234	1119	640	309	172	172
64	216583	104991	10332	1956	890	444	155	137

FIFO	PAGINAS ESCRITAS							
	128	256	512	1024	2048	4096	8192	16384
2	314	242	123	0	0	0	0	0
4	335	233	163	56	0	0	0	0
8	387	260	186	98	0	0	0	0
16	948	359	230	164	67	0	0	0
32	8827	885	277	176	134	44	0	0
64	31340	5572	867	254	149	97	5	0

matriz.log



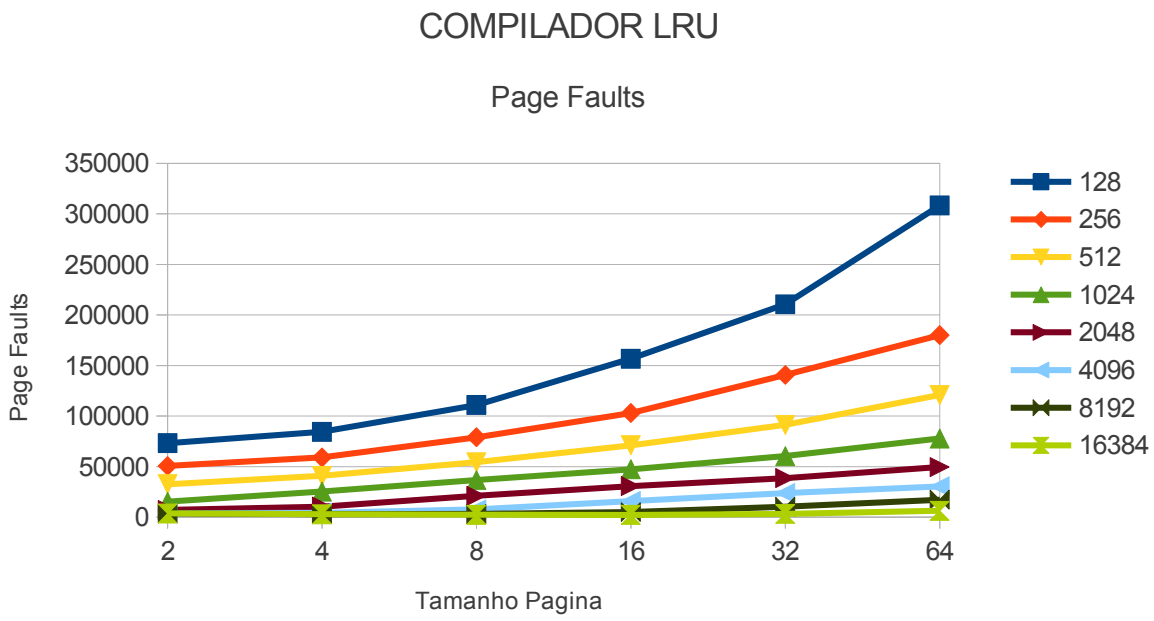


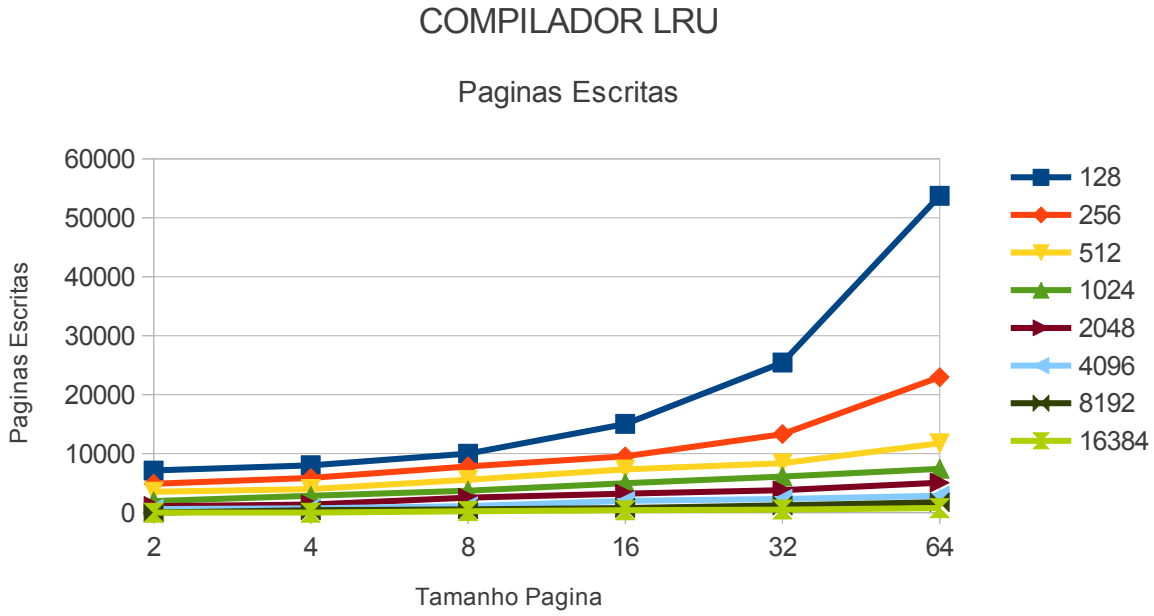
FIFO	PAGE FAUL							
	128	256	512	1024	2048	4096	8192	16384
2	36893	20441	10669	6097	4027	3367	2994	2994
4	81638	30422	17523	8551	4850	3261	2798	2543
8	211346	77190	27104	15410	6672	3840	2778	2326
16	324507	205727	70151	22815	12021	4910	3027	2294
32	304797	314398	188541	59061	19309	9506	4013	2461
64	458585	350270	271380	137751	40262	14861	6848	3175

FIFO	PAGINAS ESCRITAS							
	128	256	512	1024	2048	4096	8192	16384
2	6388	3604	2044	1281	882	436	0	0
4	14319	5492	3204	1693	1025	664	205	0
8	37940	13617	4907	2818	1311	829	538	53
16	44859	37160	12436	4170	2255	1021	692	431
32	46468	44647	34708	10287	3458	1780	847	552
64	51746	45133	45461	26232	6739	2581	1265	659

LRU

compilador.log

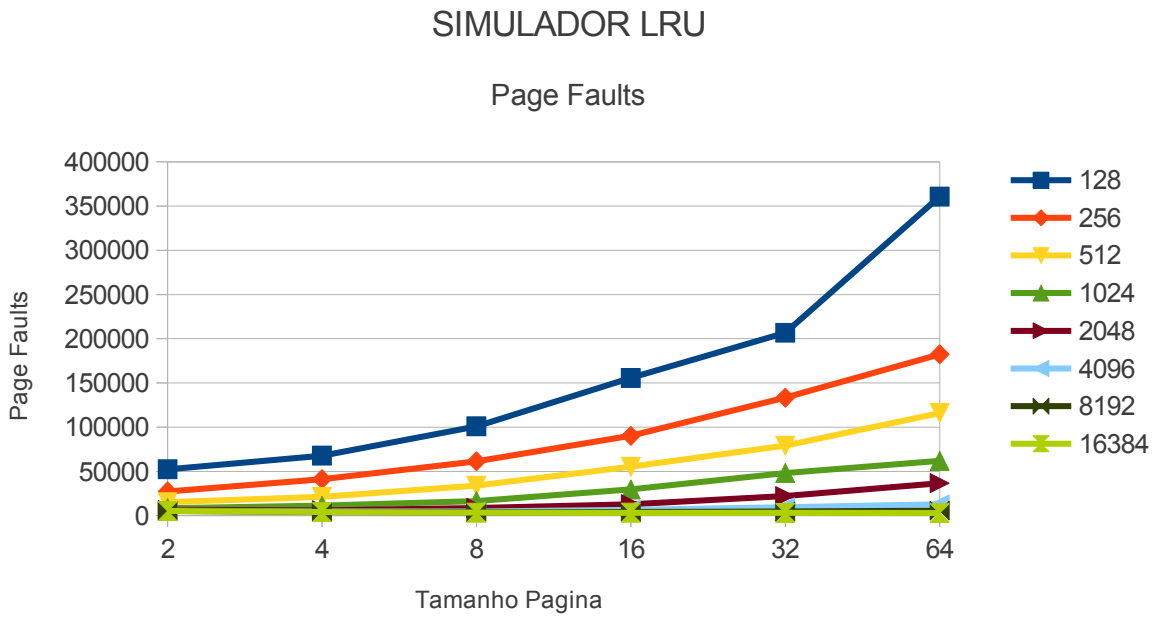


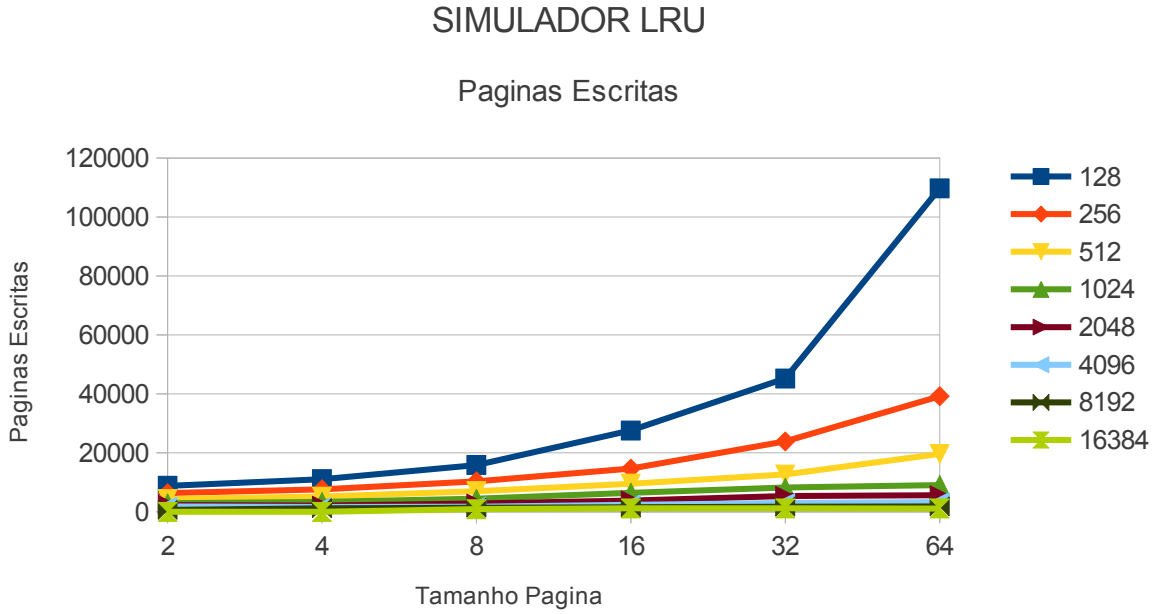


LRU	PAGE FAUL							
	128	256	512	1024	2048	4096	8192	16384
2	73200	50674	32556	15330	7074	3828	3621	3621
4	84401	59089	40821	25308	10425	4391	2904	2852
8	110909	79010	54286	36707	21091	7632	3107	2391
16	156690	102959	71099	47424	30686	15873	5084	2215
32	210633	140671	91213	60336	38641	23928	10295	3204
64	308261	179986	120815	77837	49571	30416	17265	6270

LRU	PAGINAS ESCRITAS							
	128	256	512	1024	2048	4096	8192	16384
2	7154	4923	3552	1950	1105	622	0	0
4	8039	5885	4002	2848	1384	726	399	0
8	10010	7854	5572	3728	2549	1095	573	256
16	15033	9550	7346	4995	3228	1975	797	405
32	25454	13317	8383	6137	3809	2340	1230	518
64	53732	22970	11771	7440	5089	2875	1778	771

simulador.log



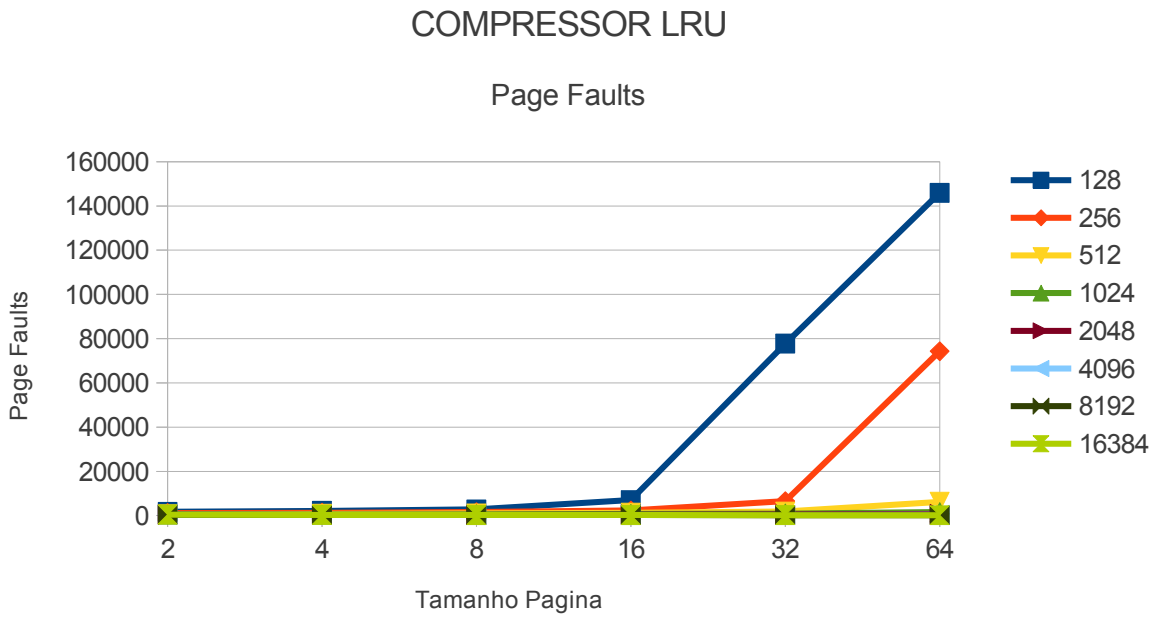


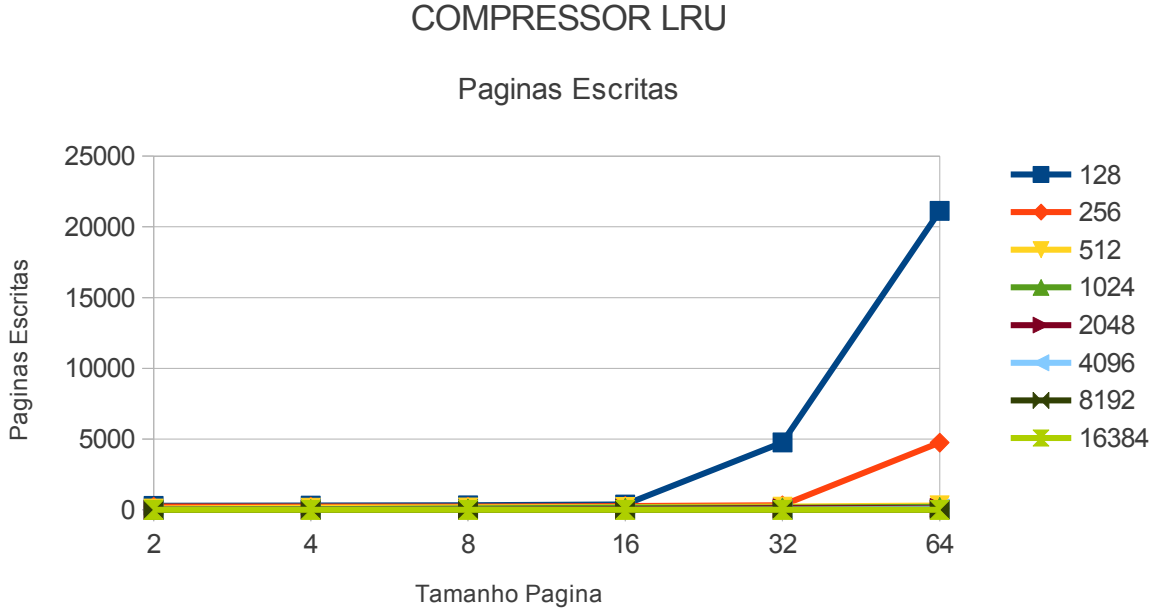
SIMULADOR

LRU	PAGE FAUL							
	128	256	512	1024	2048	4096	8192	16384
2	52359	27355	15219	8364	6250	5415	5152	5144
4	67747	41186	21090	11240	5823	4468	3951	3890
8	100880	61335	33827	16479	8556	4732	3784	3405
16	155689	90243	55174	29547	12822	6520	3924	3155
32	206695	133374	79053	48088	22126	9348	4517	2997
64	360820	182456	116031	61829	36761	12665	5510	2786

LRU	PAGINAS ESCRITAS							
	128	256	512	1024	2048	4096	8192	16384
2	8844	6338	4356	2885	2284	1803	644	0
4	11084	7638	5156	3346	2087	1624	1227	0
8	15882	10375	7027	4412	2819	1794	1403	919
16	27578	14658	9520	6452	3845	2398	1545	1170
32	45238	23888	12673	8245	5379	2953	1758	1179
64	109763	39214	19646	9088	5621	3325	1849	1127

compressor.log



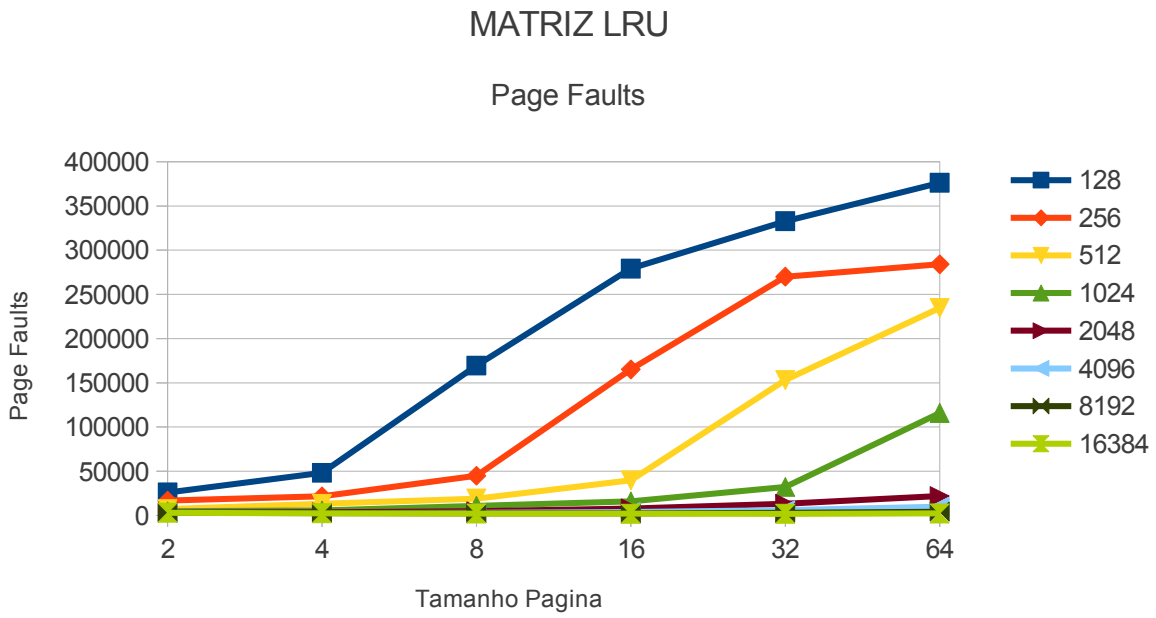


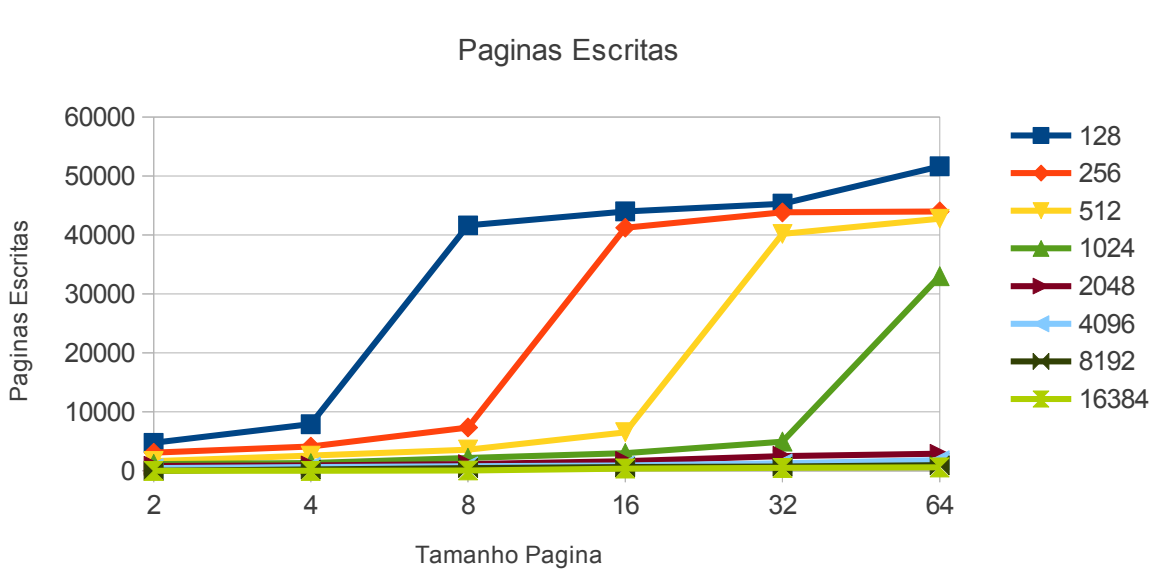
COMPRESSOR

LRU	PAGE FAUL							
	128	256	512	1024	2048	4096	8192	16384
2	1688	1134	642	419	419	419	419	419
4	2133	1264	771	397	317	317	317	317
8	2757	1558	934	533	255	255	255	255
16	7033	2302	1259	729	366	209	209	209
32	77734	6524	1811	911	515	231	172	172
64	145977	74278	6248	1566	718	357	137	137

LRU	PAGINAS ESCRITAS							
	128	256	512	1024	2048	4096	8192	16384
2	288	223	95	0	0	0	0	0
4	299	224	148	43	0	0	0	0
8	316	229	172	91	0	0	0	0
16	379	274	198	146	54	0	0	0
32	4765	319	225	155	114	31	0	0
64	21122	4747	309	206	128	83	6	0

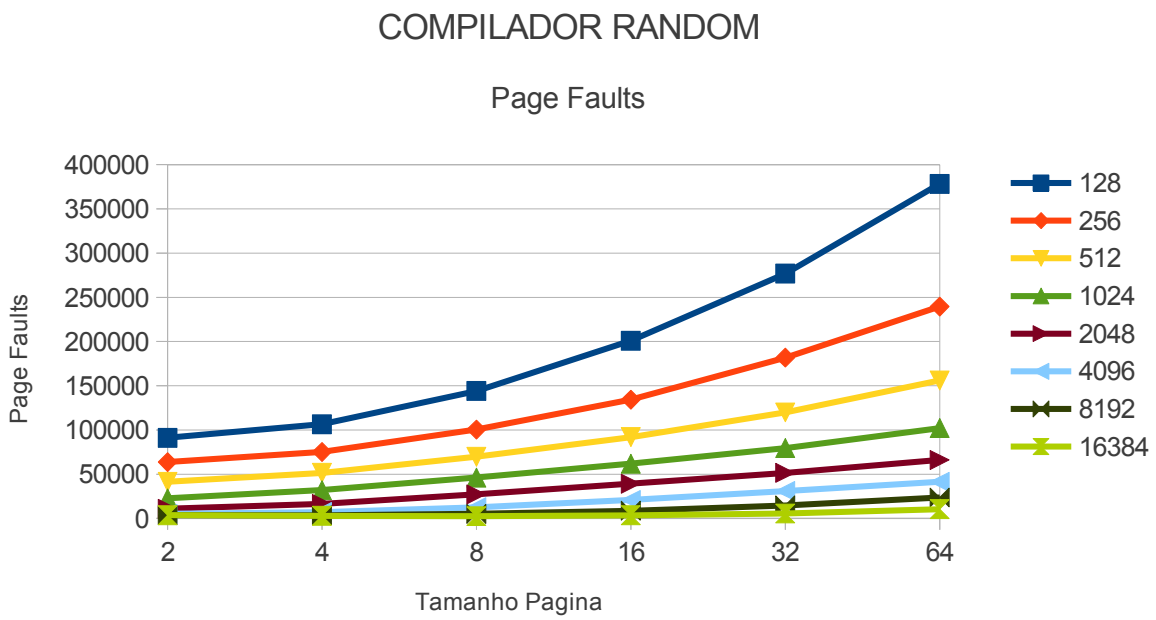
matriz.log

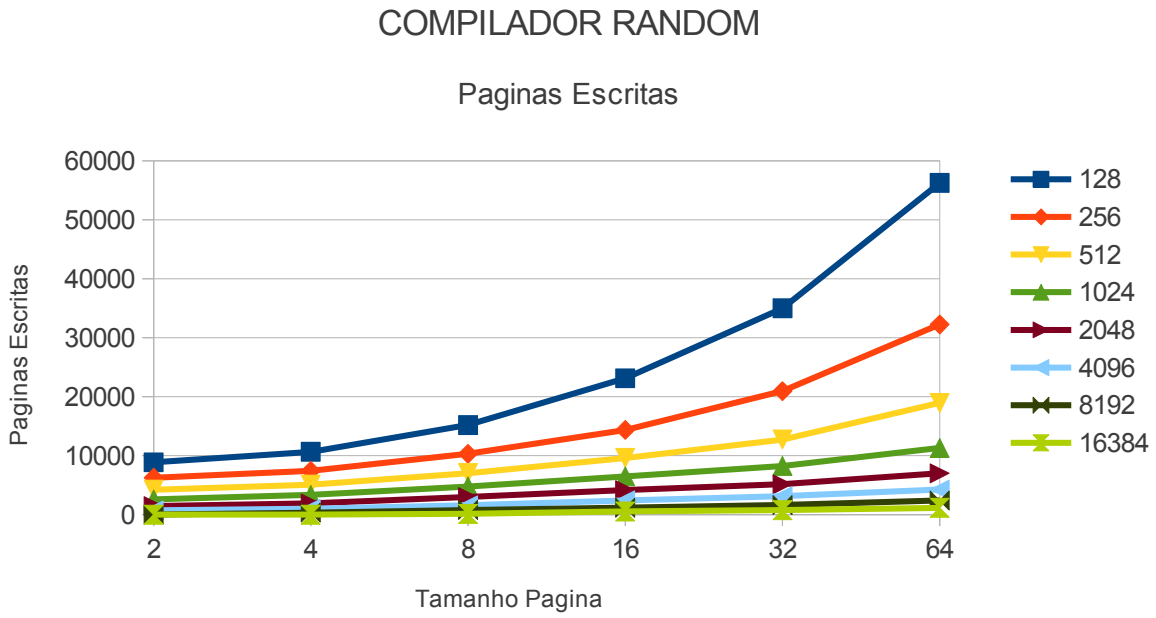




MATRIZ LRU	PAGE FAUL							
	128	256	512	1024	2048	4096	8192	16384
2	26003	16791	7172	4744	3373	3061	2994	2994
4	48254	21656	13250	5673	3632	2803	2576	2543
8	169540	44904	19141	10928	4745	2950	2398	2214
16	279236	165115	39728	15904	7876	3828	2380	2011
32	332851	270086	153176	32282	13230	5974	2961	1950
64	376284	284027	234688	115886	21930	9963	4328	2331
LRU	PAGINAS ESCRITAS							
	128	256	512	1024	2048	4096	8192	16384
2	4748	3080	1660	1163	835	436	0	0
4	7903	4128	2585	1317	923	643	229	0
8	41630	7356	3601	2194	1125	777	487	64
16	43997	41220	6539	3002	1665	912	636	375
32	45309	43835	40184	4954	2501	1282	725	496
64	51640	43993	42763	33029	2907	1799	953	585

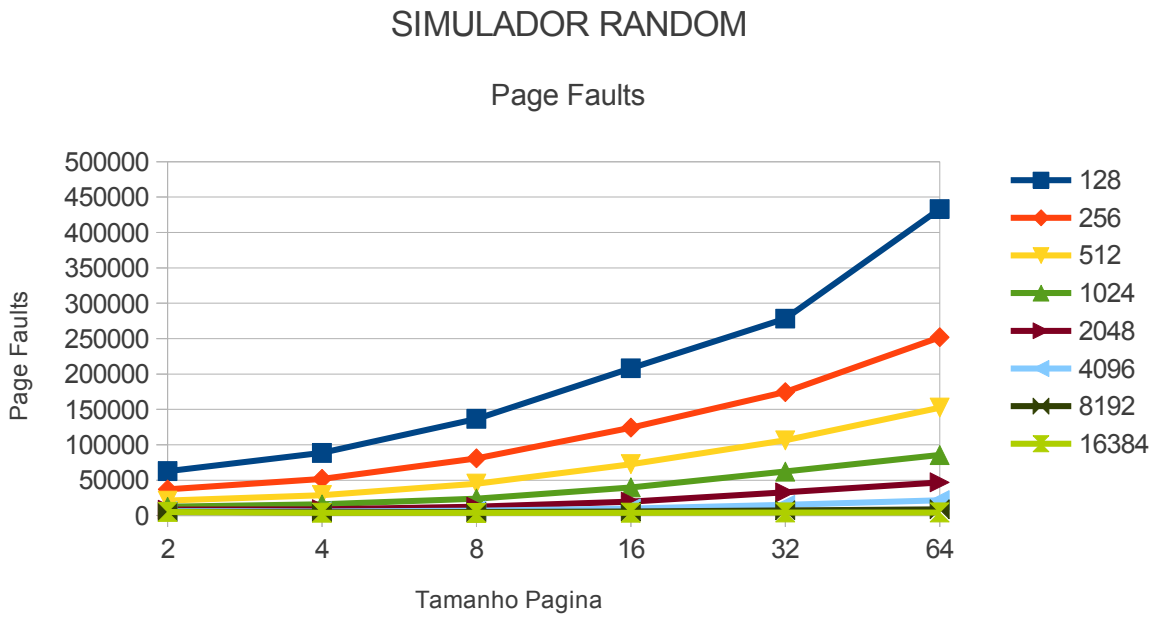
RANDOM
compilador.log

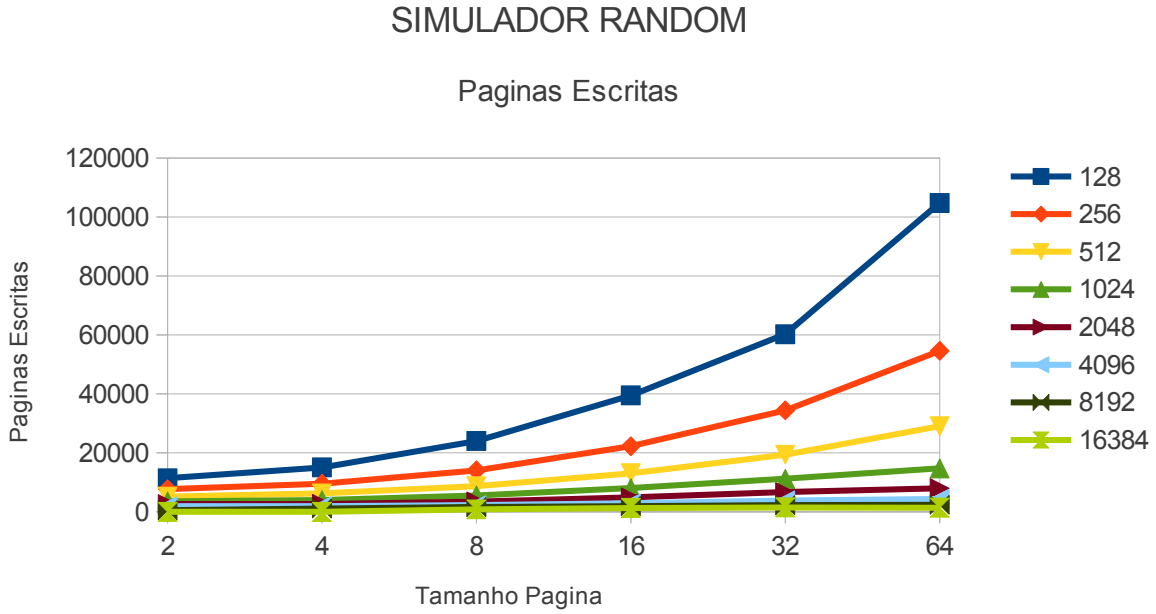




COMPILADOR RANDOM	PAGE FAUL							
	128	256	512	1024	2048	4096	8192	16384
2	91118	63697	41368	22624	10871	5182	3621	3621
4	106522	75116	51496	32022	16291	7151	3422	2852
8	144007	100470	69677	46247	27178	12605	5123	2529
16	200908	134197	91899	61856	39198	20817	8676	3441
32	276743	181763	119875	79584	51441	30893	14673	5545
64	378345	239465	156045	102135	66083	41335	23629	10411
RANDOM	PAGINAS ESCRITAS							
	128	256	512	1024	2048	4096	8192	16384
2	8879	6264	4230	2568	1456	739	0	0
4	10646	7437	5085	3357	1891	1000	336	0
8	15207	10316	7038	4781	2989	1587	801	133
16	23126	14370	9623	6493	4167	2435	1179	547
32	35006	20952	12727	8249	5155	3132	1626	765
64	56254	32248	18941	11315	7027	4264	2397	1142

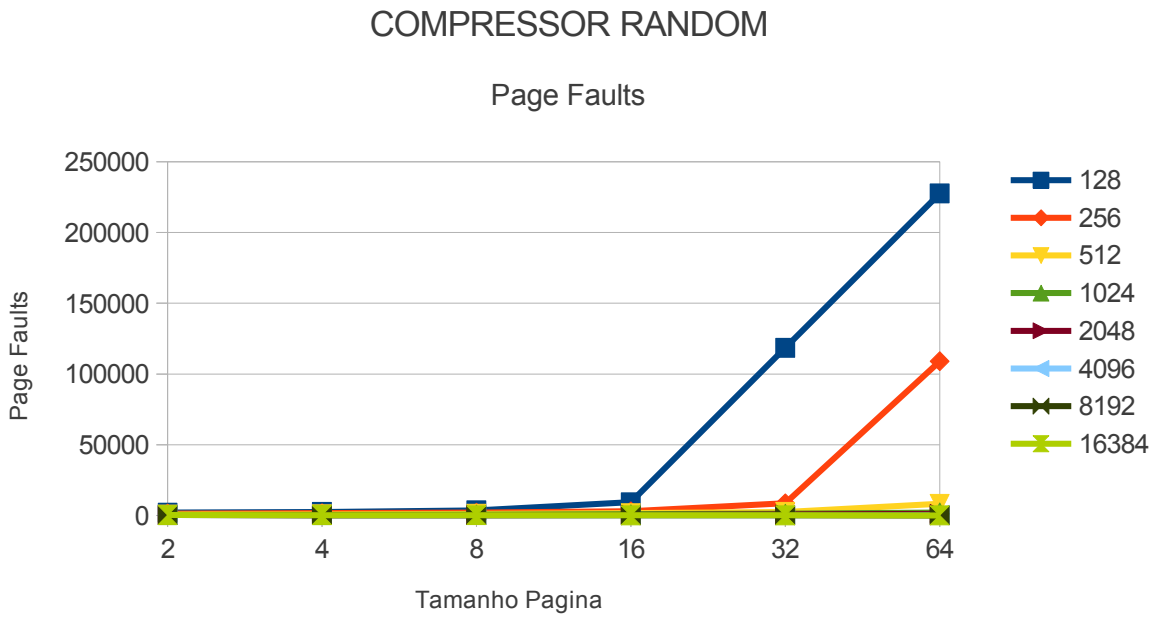
simulador.log

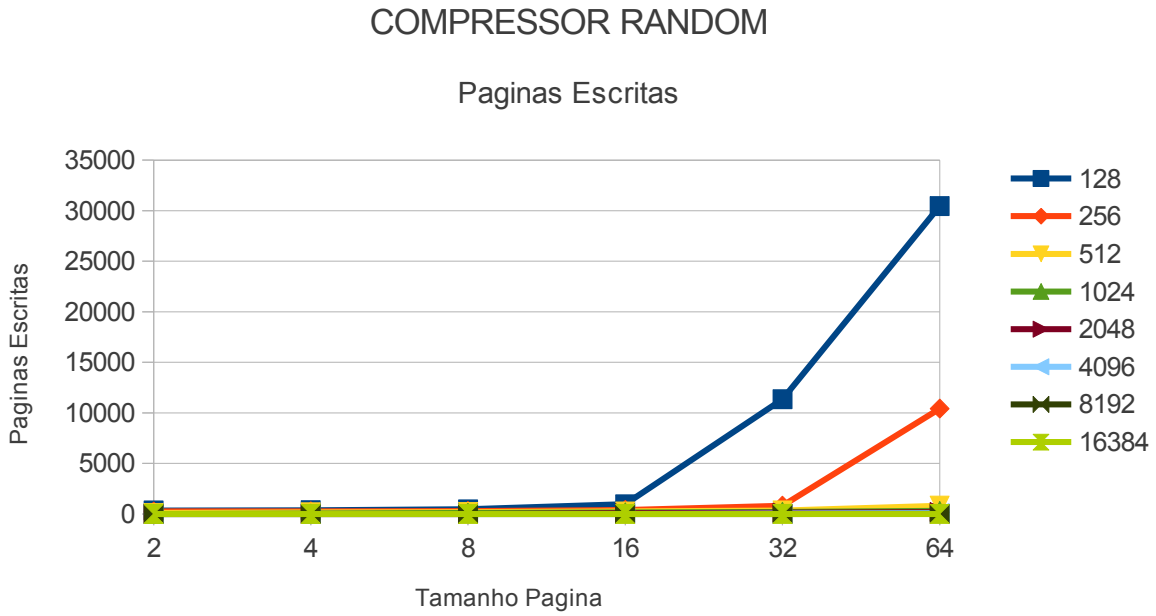




SIMULADOR RANDOM	PAGE FAUL							
	128	256	512	1024	2048	4096	8192	16384
2	62797	36907	21264	12433	8318	6344	5307	5144
4	88596	51830	28950	15779	8952	5913	4515	3890
8	136696	80917	45065	23931	12482	7019	4844	3756
16	208203	124207	72655	39849	19729	9995	5711	3960
32	278326	174522	106342	62165	32736	15131	7354	4302
64	433261	252032	152415	85620	46913	21676	8851	4337
RANDOM	PAGINAS ESCRITAS							
	128	256	512	1024	2048	4096	8192	16384
2	11421	7672	5168	3547	2634	1894	544	0
4	15076	9542	6210	3998	2672	1910	1153	0
8	24007	14078	8714	5570	3463	2241	1637	834
16	39494	22237	13083	8044	4875	2917	1929	1324
32	60252	34437	19380	11222	6699	3826	2266	1465
64	104754	54580	29027	14744	8020	4367	2367	1406

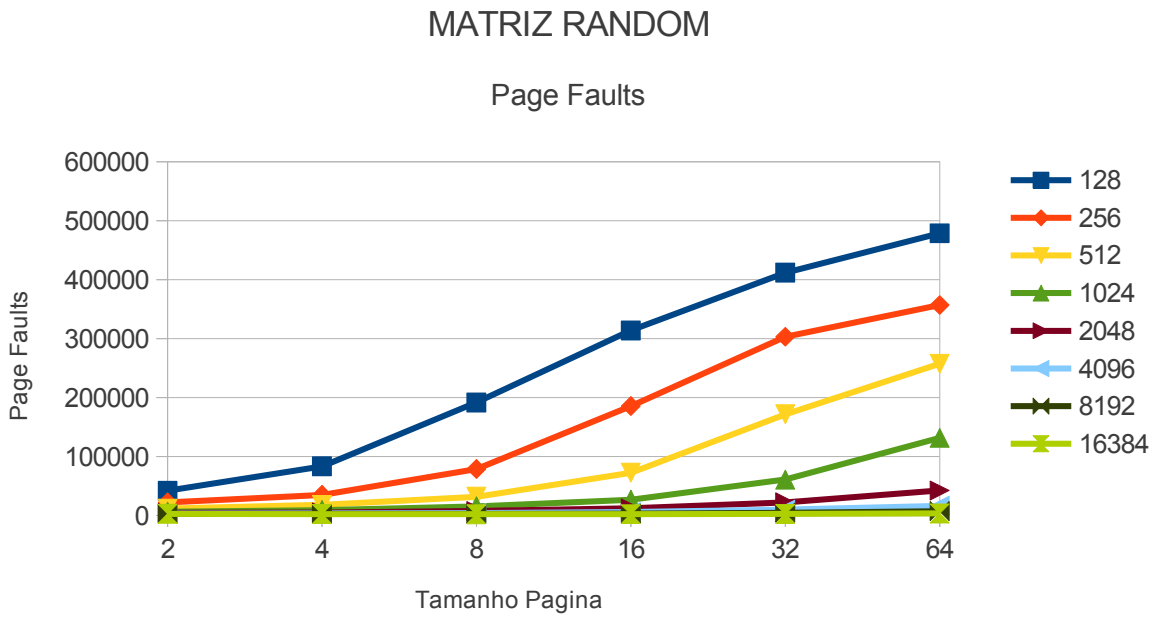
compressor.log

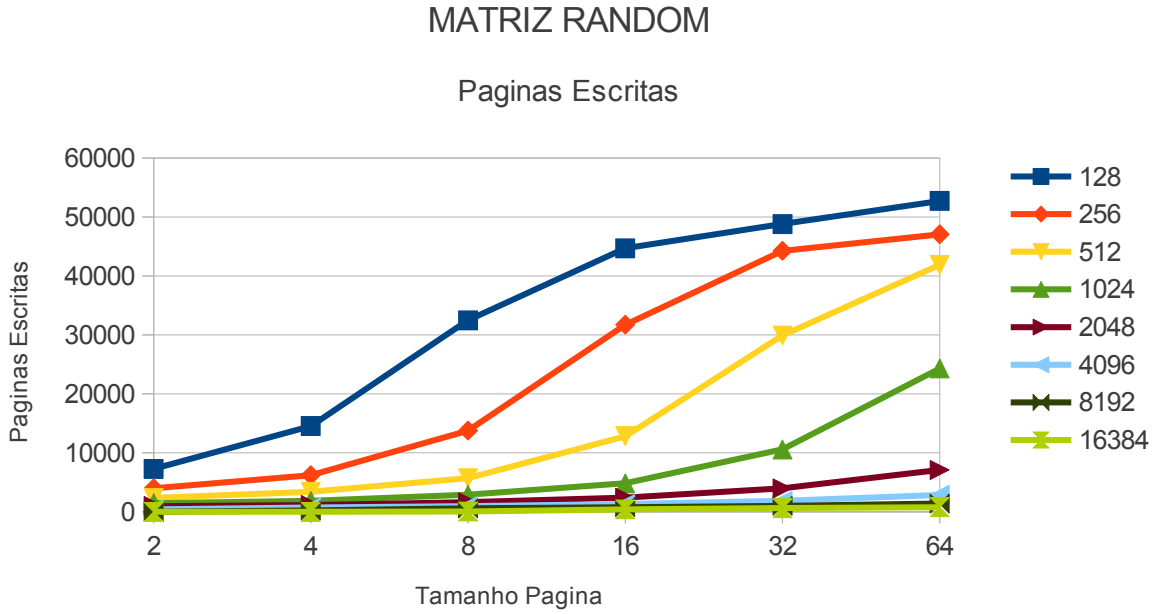




COMPRESSOR RANDOM	PAGE FAUL							
	128	256	512	1024	2048	4096	8192	16384
2	2130	1390	742	419	419	419	419	419
4	2588	1674	899	399	317	317	317	317
8	3706	2055	1197	661	255	255	255	255
16	9411	3068	1719	975	457	209	209	209
32	118509	8701	2586	1328	709	268	172	172
64	227610	109066	8294	2167	1090	475	138	137
RANDOM	PAGINAS ESCRITAS							
	128	256	512	1024	2048	4096	8192	16384
2	331	244	95	0	0	0	0	0
4	360	258	155	37	0	0	0	0
8	457	278	183	101	0	0	0	0
16	956	387	249	153	61	0	0	0
32	11339	813	332	202	120	30	0	0
64	30460	10415	811	295	176	92	3	0

matriz.log





MATRIZ RANDOM	PAGE FAUL							
	128	256	512	1024	2048	4096	8192	16384
2	42167	22496	11899	6647	4370	3291	2994	2994
4	83497	35009	18141	9260	5239	3474	2662	2543
8	191914	79045	31949	15523	7742	4333	2928	2238
16	313974	185803	72700	26967	12269	5983	3402	2312
32	411959	303148	171706	61009	22545	9900	4679	2730
64	478731	357040	257464	131764	42514	16609	7442	3676
RANDOM	PAGINAS ESCRITAS							
	128	256	512	1024	2048	4096	8192	16384
2	7293	4009	2340	1473	972	415	0	0
4	14554	6223	3410	1866	1142	716	225	0
8	32470	13780	5711	2887	1621	985	586	59
16	44677	31740	12828	4829	2390	1307	790	415
32	48817	44256	29909	10585	3978	1850	1012	621
64	52720	47064	41886	24308	7104	2860	1396	799

BUG e Decisões de Projeto

Aparentemente não foram encontrados bugs no programa. Porém o programa em si não trata falha de linhas de execução sem parametro. Ele trata apenas a falta do parametro para debug , nesse caso não é apresentado as linhas de debug.

Aproveitando , para debugar o programa é simples , basta adicionar um parametro ao final da linha de execução utilizando 0 ou 1 . No caso 0 indica sem debug , 1 indica o debug . Esse mostra qual registro vai ser inserido e como é o status da memória , nesse caso quais paginas estão alocadas na memória , qual o tempo que ela foi alocada e qual é o tipo de permissão de acesso (“R” ou “W”) .

Em caso de ativação do modo debug o programa que executa em segundos tem um aumento drastico para horas , isso porque para cada inserção é mostrado o vetor da memória , e percorre-lo todo é custosso.