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 .

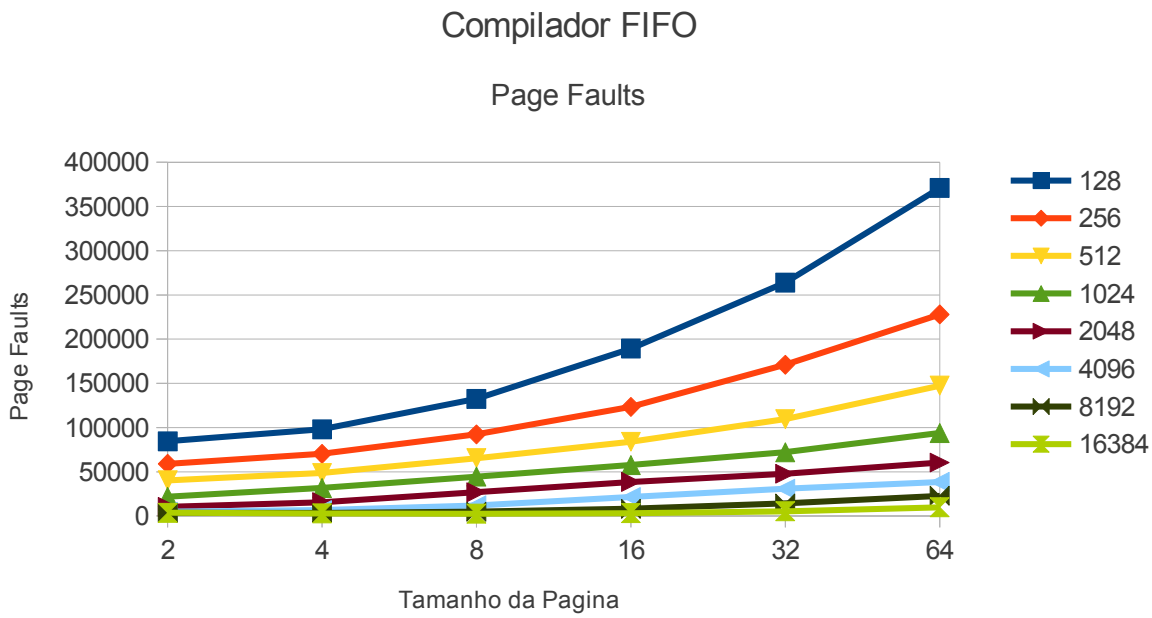
Conclusão geral

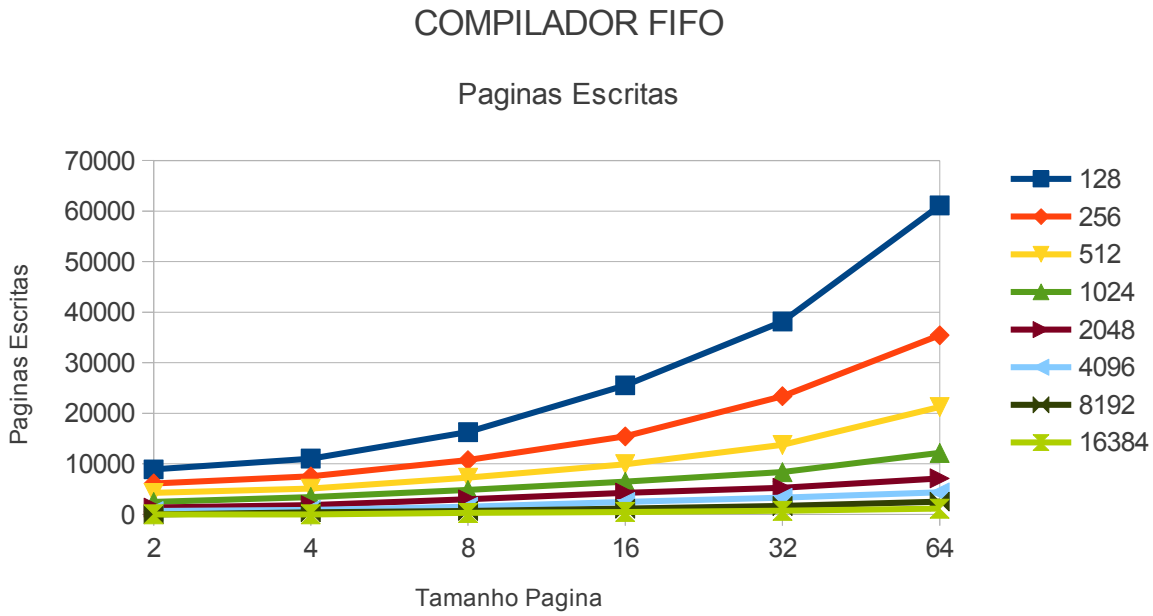
Após a analise dos graficos abaixo e da serie de testes foi possível perceber que o metodo de reposição de paginas LRU é o mais eficiente , não apenas por questões de page faults , mas também pela retirada de paginas escritas . Se considerarmos que em alguns casos essas paginas escritas não foram retiradas 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



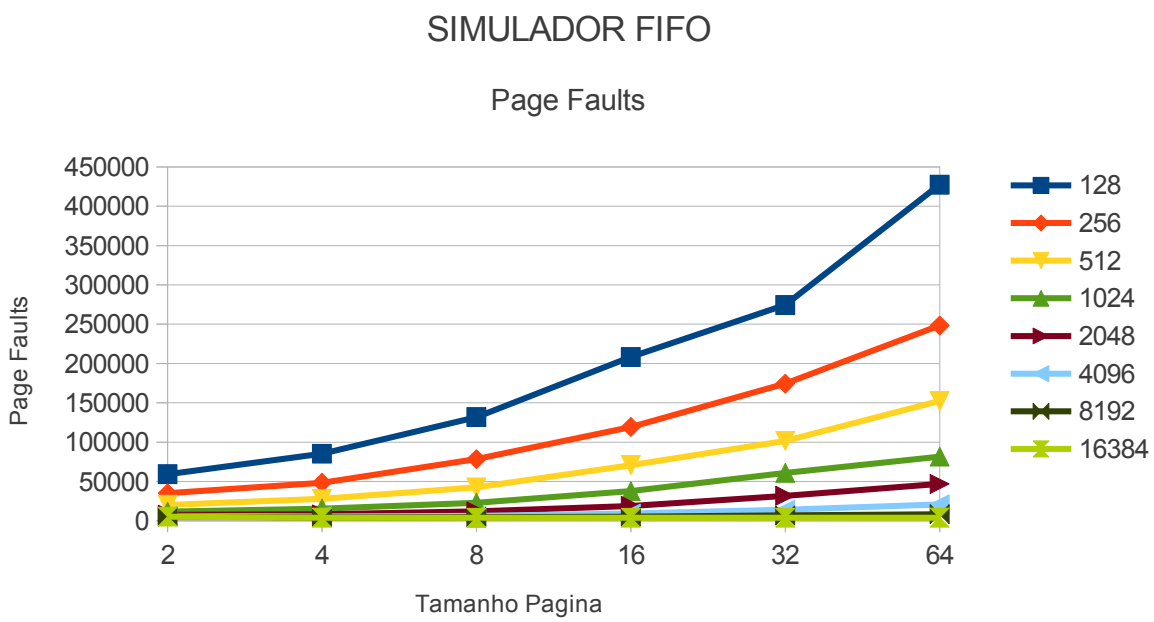


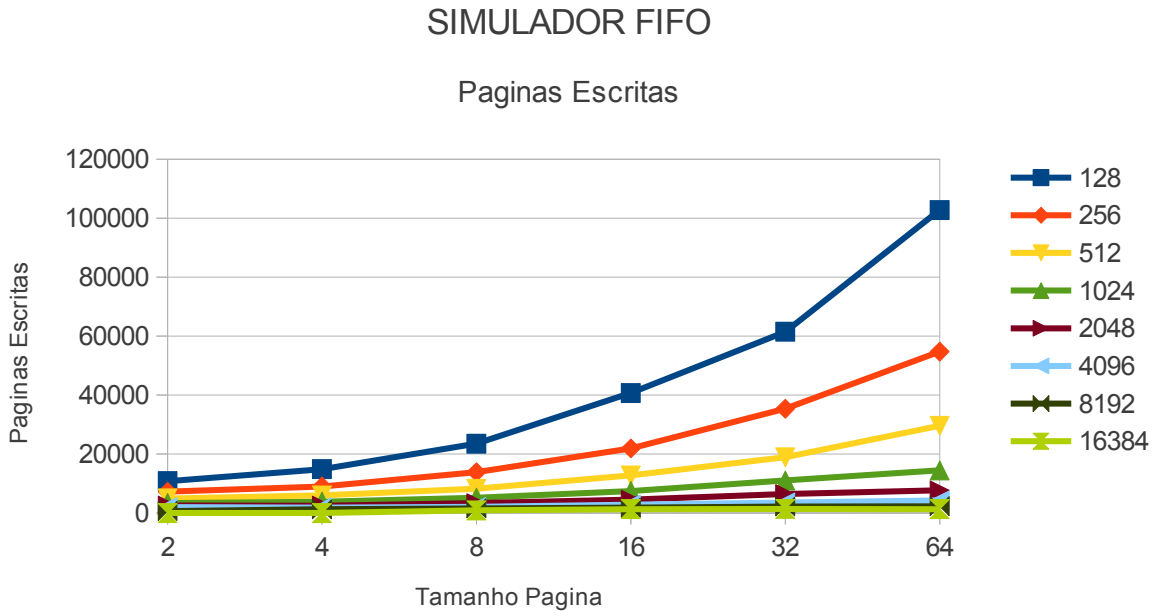
COMPILADOR

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	123445	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	8896	6112	4235	2426	1331	631	0	0
4	11025	7512	5081	3422	1821	878	418	0
8	16271	10718	7236	4807	2966	1496	676	276
16	25517	15415	9907	6447	4253	2426	1108	468
32	38152	23359	13723	8344	5228	3301	1632	694
64	61107	35426	21218	12150	7110	4353	2511	1132

simulador.log

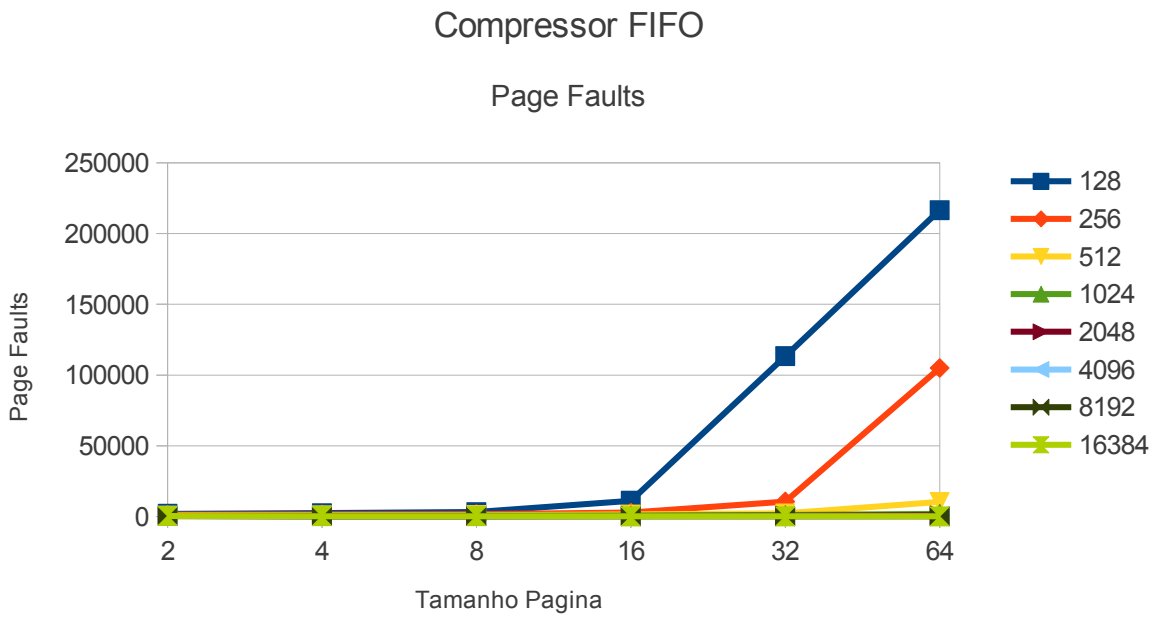


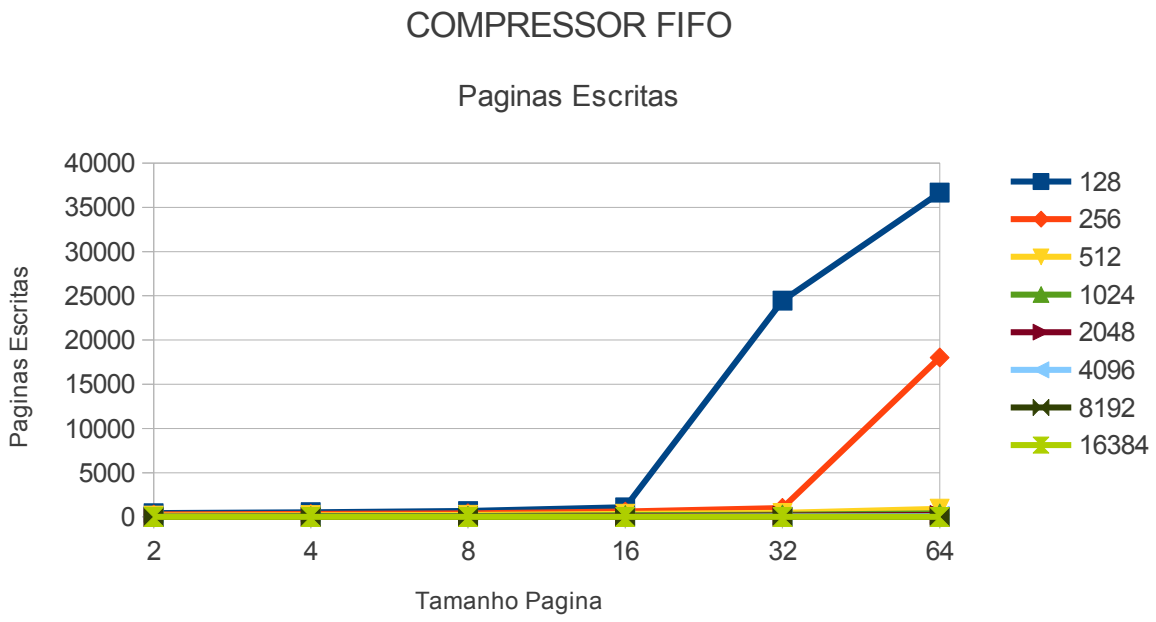


SIMULADOR FIFO	PAGE FAUL							
	128	256	512	1024	2048	4096	8192	16384
2	59155	34943	20163	11449	7663	6168	5333	5144
4	85283	48301	27778	15440	8089	5492	4314	3890
8	131827	78338	42421	22816	11884	6293	4528	3669
16	208367	119307	70724	37606	18800	9188	5108	3751
32	274439	174434	101483	60922	31700	14066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

FIFO	PAGINAS ESCRITAS							
	128	256	512	1024	2048	4096	8192	16384
2	10812	7237	4921	3394	2533	1953	687	0
4	14870	8979	5976	3860	2486	1844	1263	0
8	23530	13864	8239	5184	3330	2101	1576	898
16	40706	21908	12728	7479	4536	2831	1806	1285
32	61494	35348	18918	11045	6415	3579	2200	1381
64	102791	54716	29562	14469	7718	4237	2283	1323

compressor.log



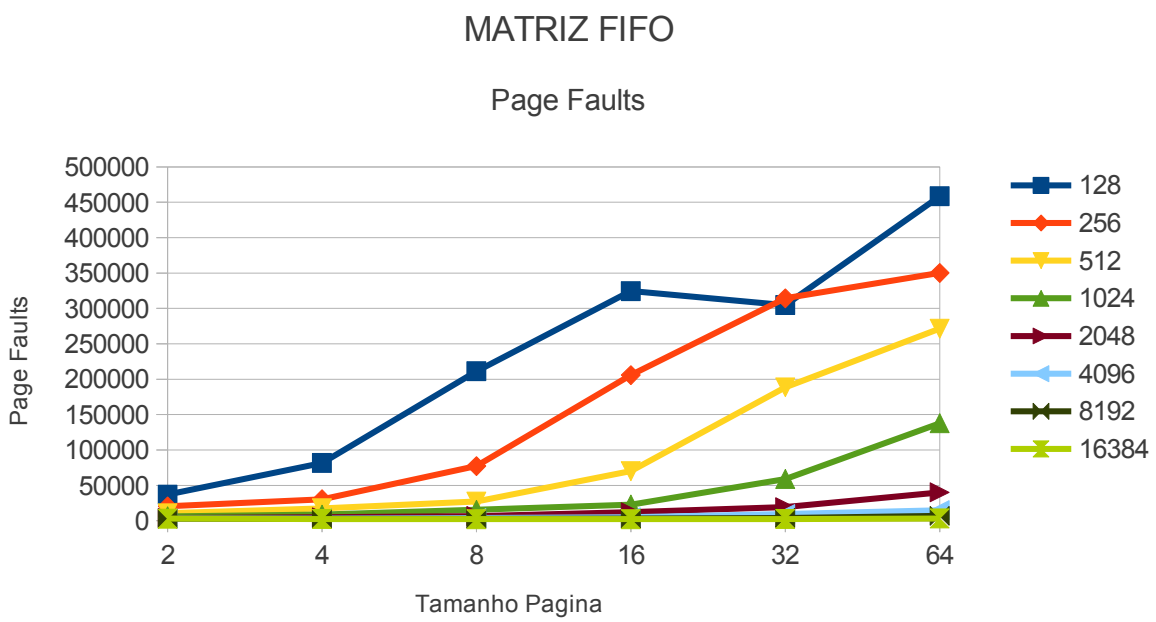


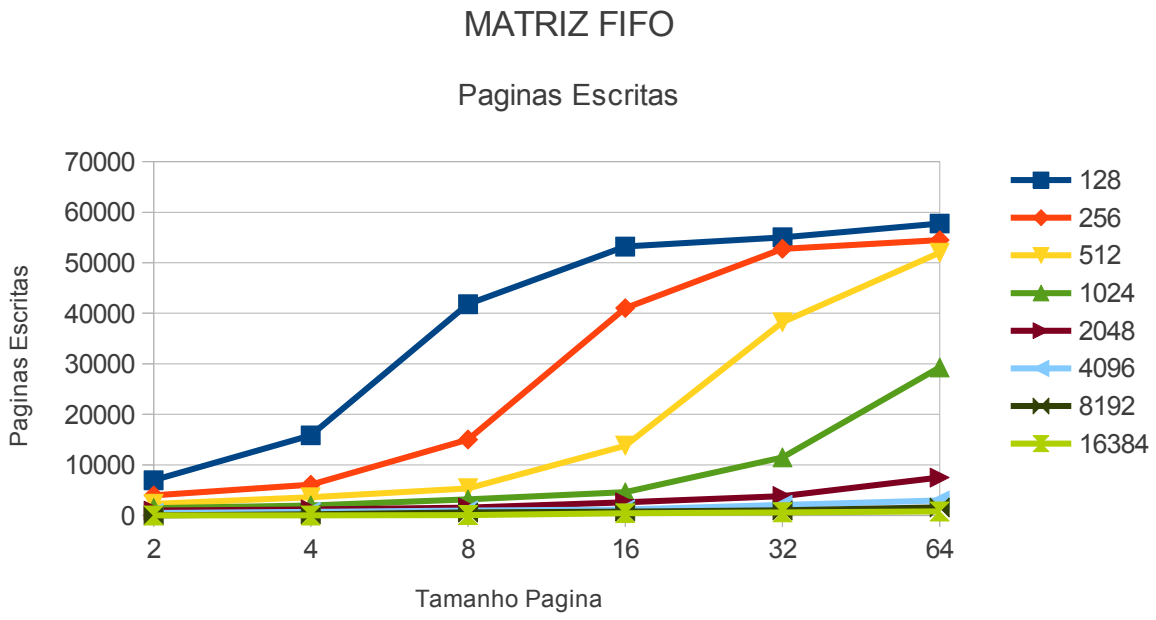
COMPRESSOR

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	308	172	172
64	216584	104991	10332	1956	890	444	155	137

FIFO	PAGINAS ESCRITAS							
	128	256	512	1024	2048	4096	8192	16384
2	434	279	152	0	0	0	0	0
4	525	338	202	94	0	0	0	0
8	675	403	236	118	0	0	0	0
16	1080	612	347	200	85	0	0	0
32	24448	1009	452	258	150	69	0	0
64	36662	18021	930	396	212	113	5	0

matriz.log

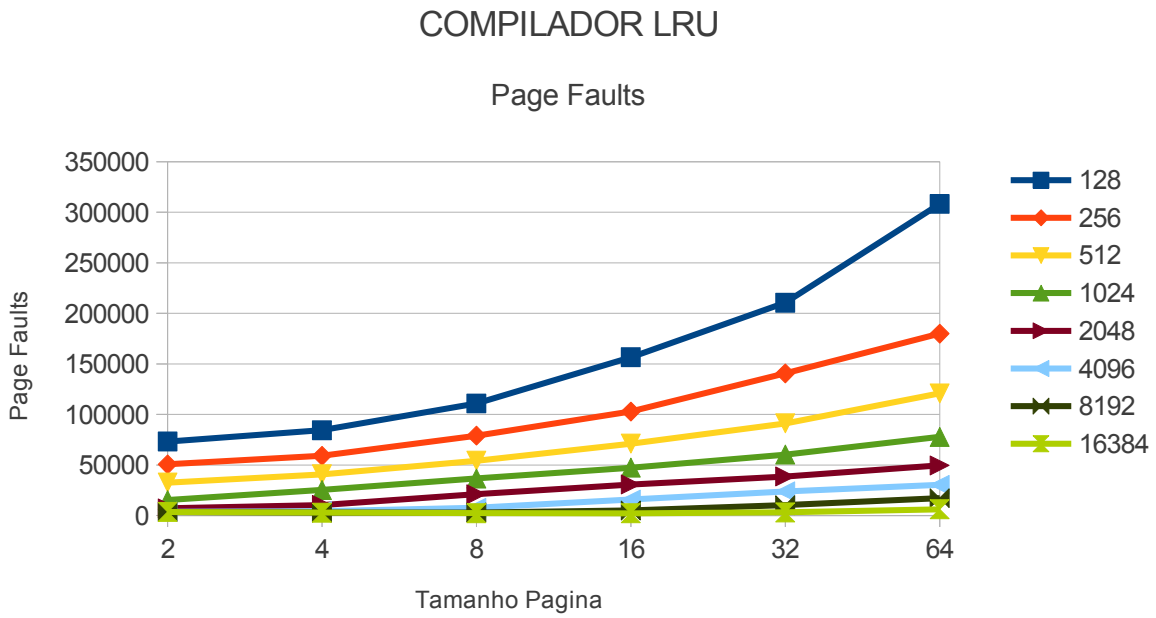


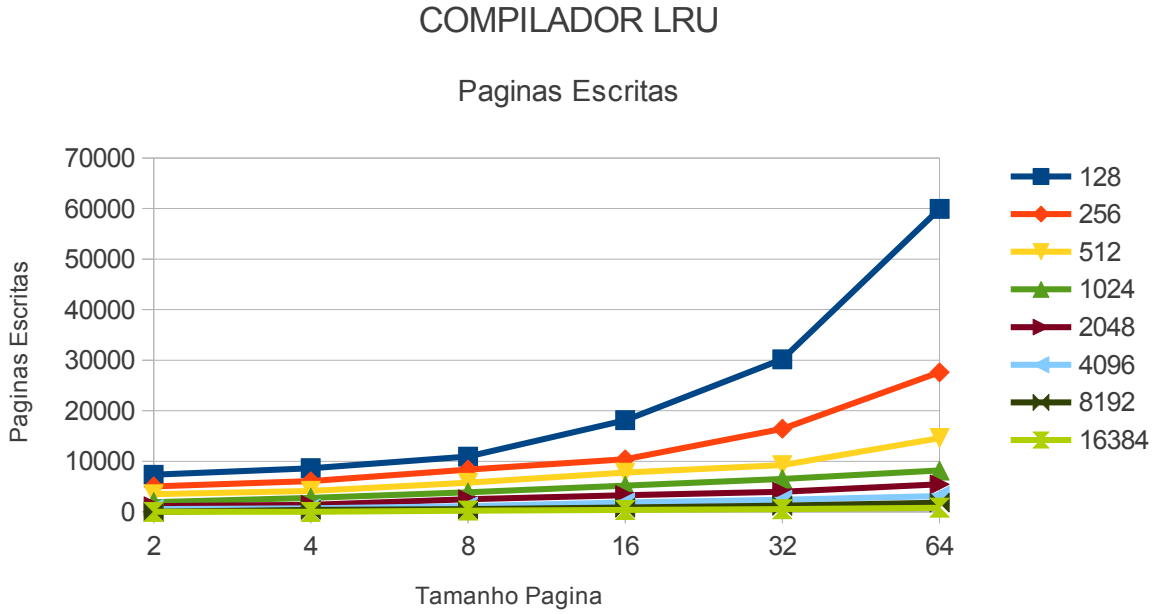


MATRIZ FIFO	PAGE FAUL							
	128	256	512	1024	2048	4096	8192	16384
2	36893	20441	10669	6097	4027	3367	2994	2994
4	81638	30422	17523	8551	4850	3264	2798	2543
8	211346	77190	27104	15410	6672	3840	2778	2326
16	324507	205727	70151	22815	12021	4910	3027	2294
32	403797	314398	188541	59061	19309	9506	4013	2461
64	458585	350270	271380	137751	40262	14861	6848	3175
FIFO	PAGINAS ESCRITAS							
	128	256	512	1024	2048	4096	8192	16384
2	6965	3962	2300	1455	957	470	0	0
4	15845	6059	3589	1985	1184	724	230	0
8	41805	15031	5365	3178	1561	978	592	77
16	53210	41023	13801	4578	2602	1199	771	446
32	55019	52752	38229	11478	3834	2109	991	613
64	57748	54482	51987	29304	7482	2975	1573	797

LRU

compilador.log



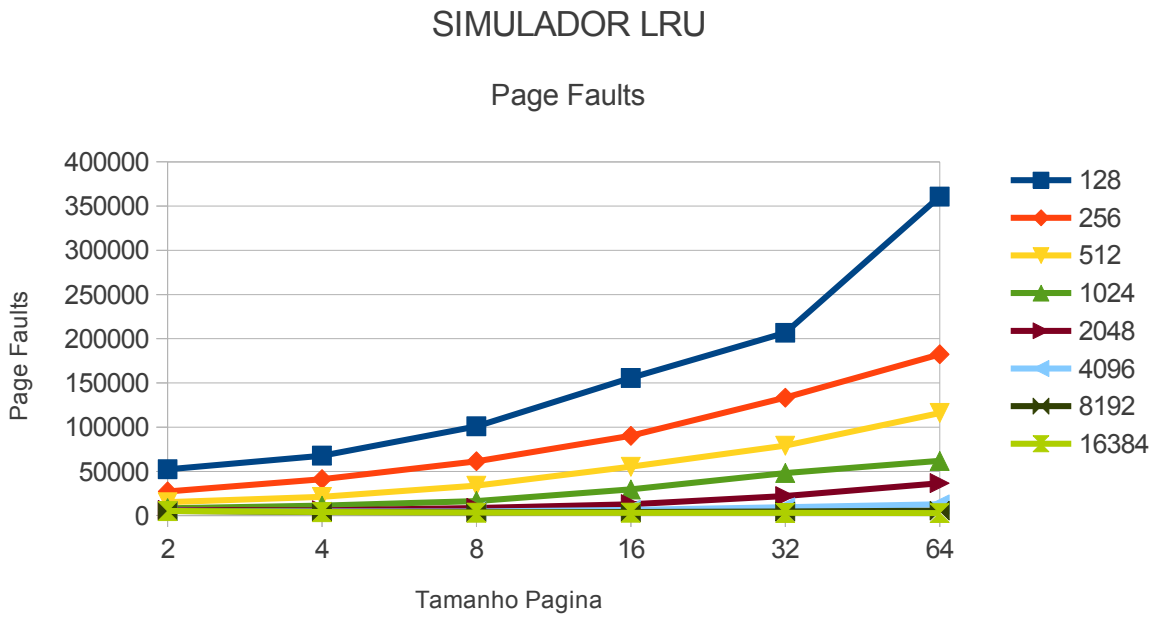


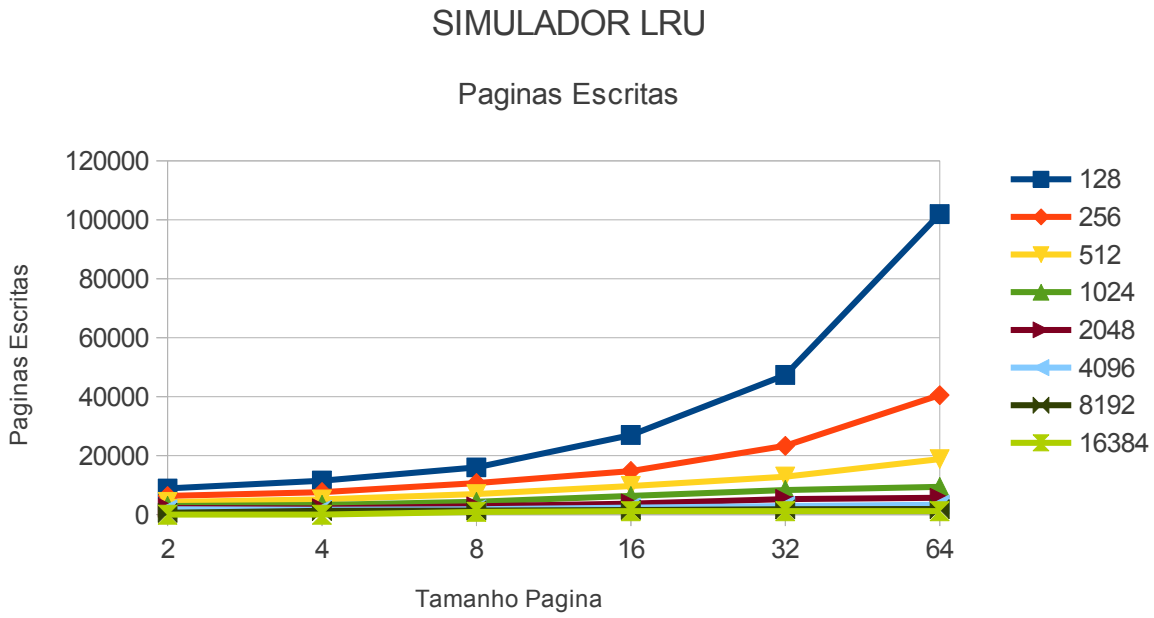
COMPILADOR

RANDOM	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

RANDOM	PAGINAS ESCRITAS							
	128	256	512	1024	2048	4096	8192	16384
2	7326	5041	3475	1935	1101	549	0	0
4	8647	6082	4131	2776	1389	693	375	0
8	10911	8368	5767	3860	2476	1122	520	251
16	18147	10410	7761	5162	3298	1882	829	365
32	30179	16430	9226	6496	3978	2403	1222	534
64	59936	27598	14601	8176	5424	3138	1806	816

simulador.log



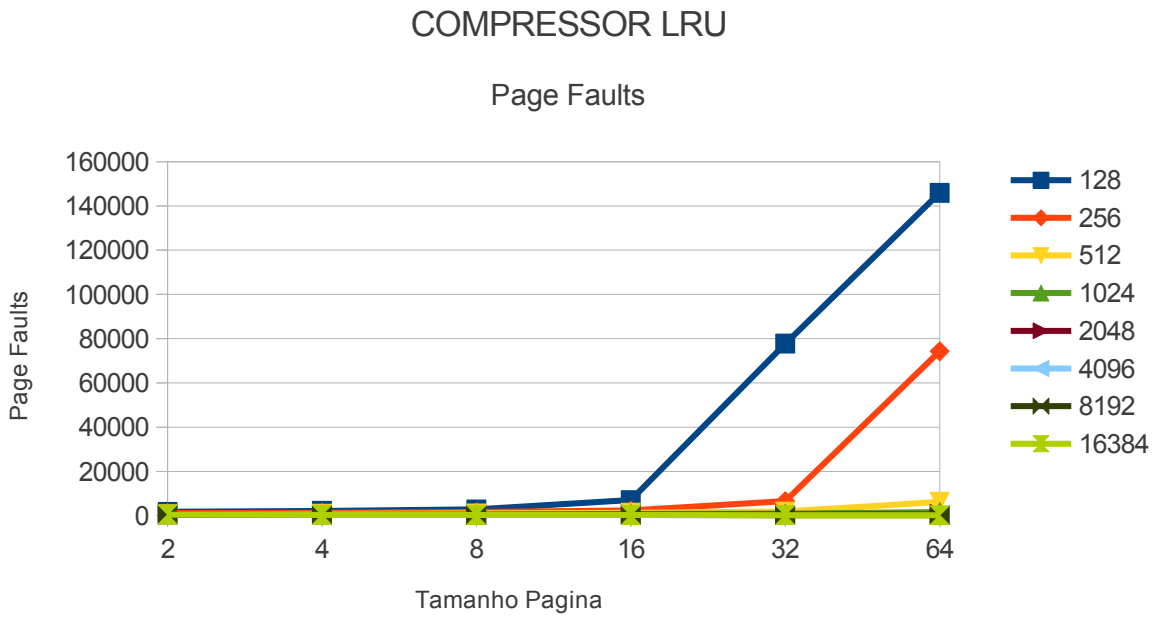


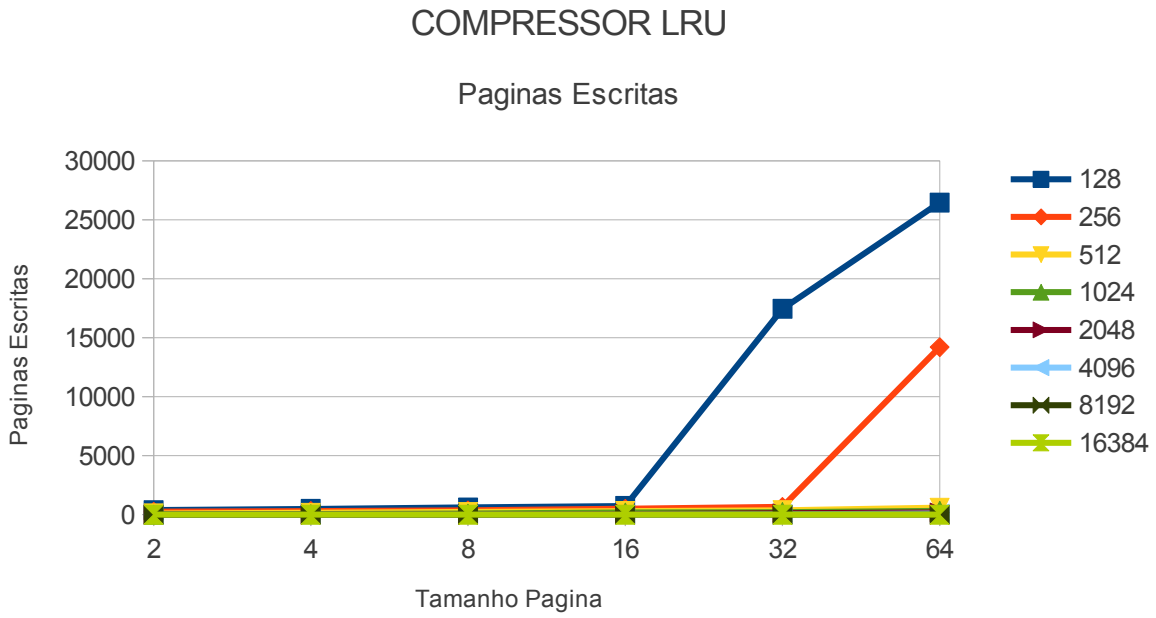
SIMULADOR

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	8872	6317	4333	2930	2314	1821	630	0
4	11553	7646	5138	3325	2121	1652	1261	0
8	16032	10703	6974	4347	2822	1829	1425	930
16	27006	14788	9725	6343	3749	2390	1571	1177
32	47345	23357	12887	8365	5306	2938	1820	1197
64	101966	40527	18826	9391	5675	3389	1886	1169

compressor.log

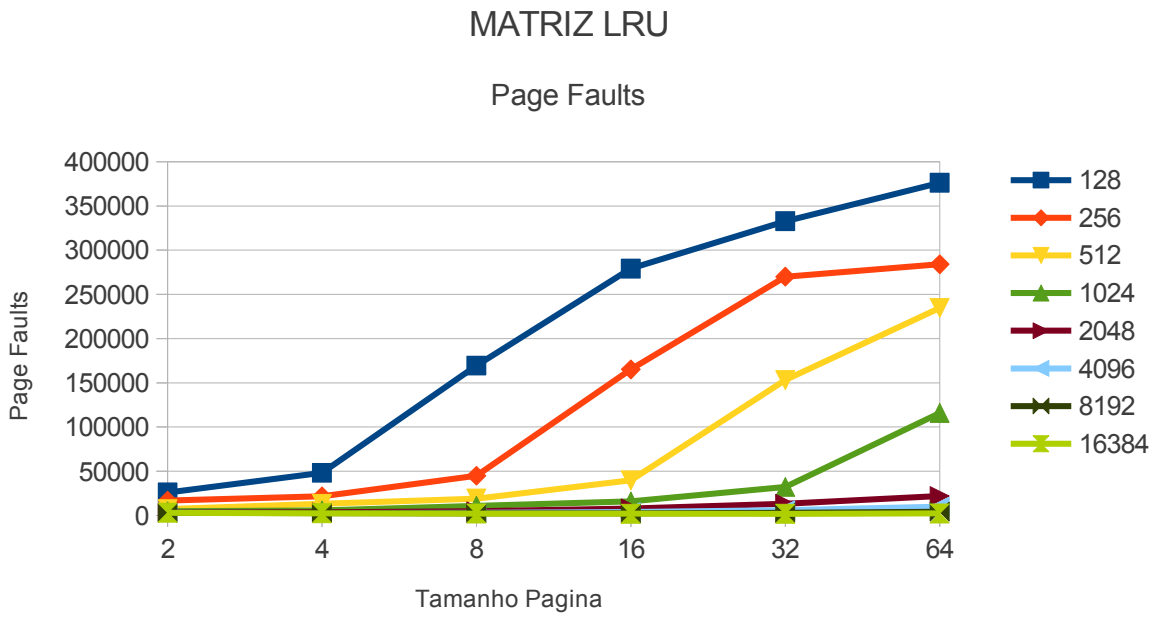


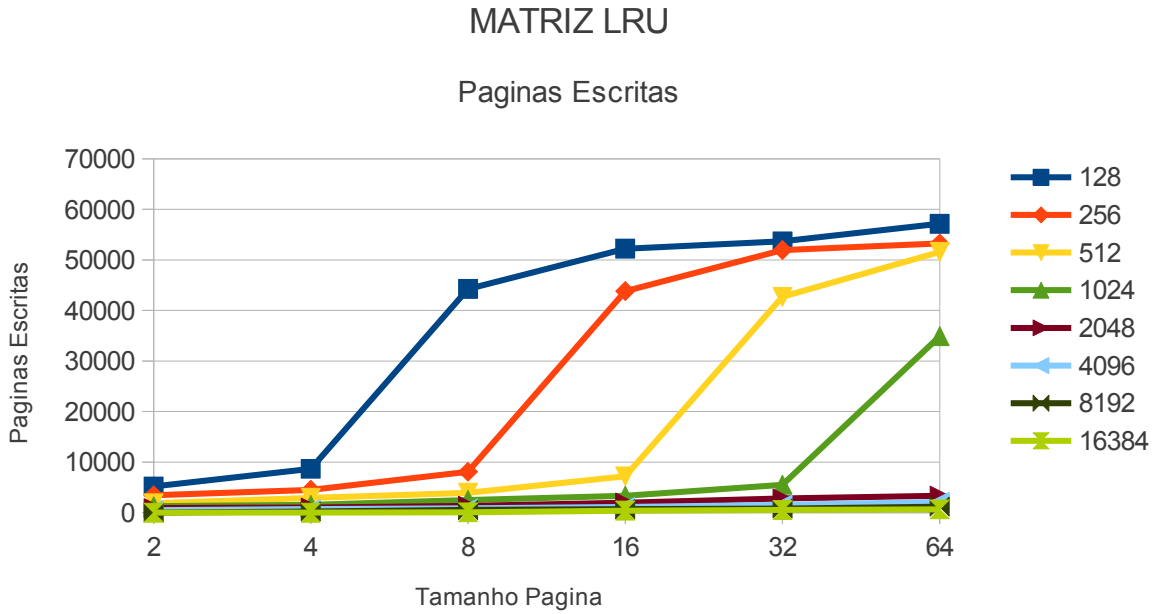


COMPRESSOR LRU	PAGE FAUL							
	128	256	512	1024	2048	4096	8192	16384
2	1688	1134	642	419	419	419	419	419
4	2133	1264	771	397	317	317	317	317
8	2757	1558	934	533	255	255	255	255
16	7033	2302	1259	729	366	209	209	209
32	77734	6524	1811	911	515	231	172	172
64	145977	74278	6248	1566	718	357	137	137

LRU	PAGINAS ESCRITAS							
	128	256	512	1024	2048	4096	8192	16384
2	385	250	102	0	0	0	0	0
4	484	302	165	48	0	0	0	0
8	610	369	225	90	0	0	0	0
16	732	502	309	185	62	0	0	0
32	17456	654	375	215	119	36	0	0
64	26467	14205	595	332	176	84	6	0

matriz.log

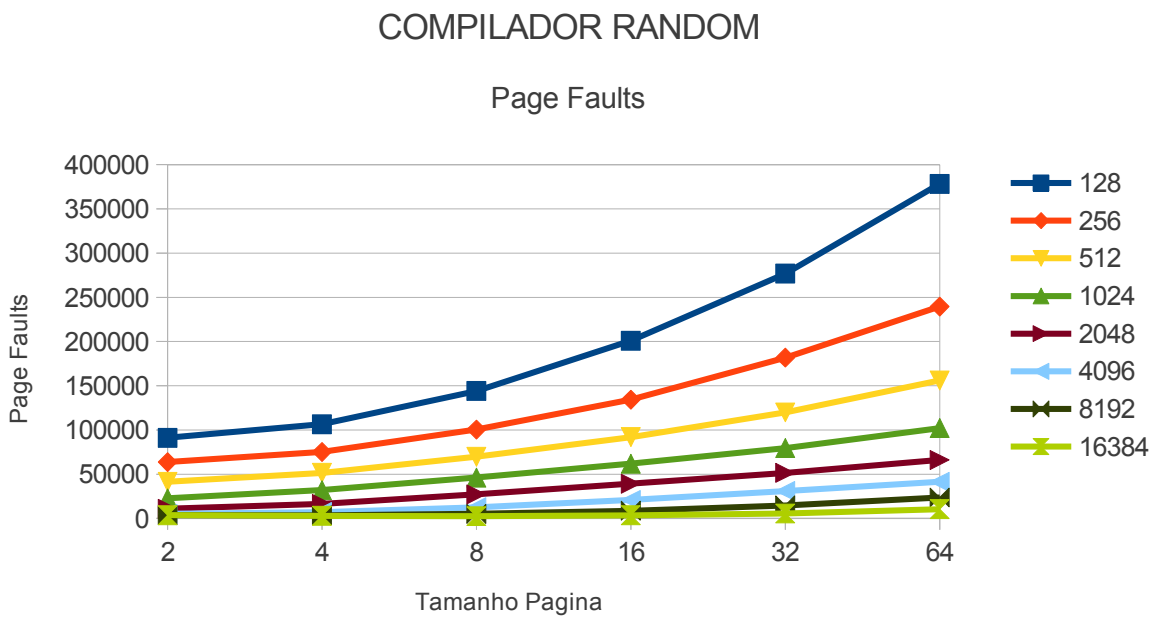


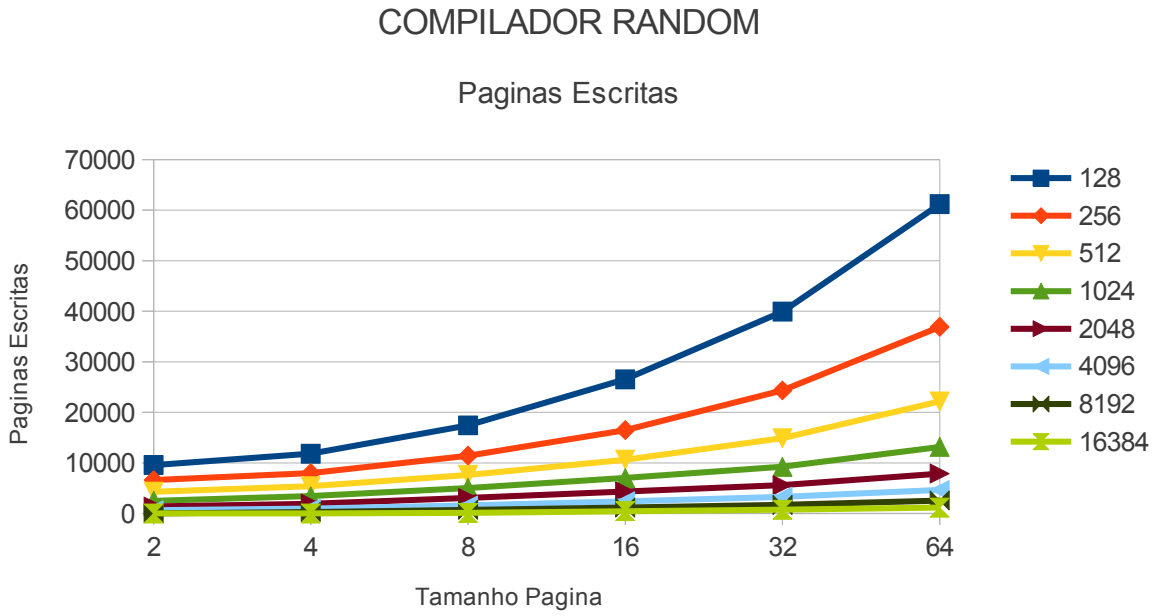


MATRIZ LRU	PAGE FAUL							
	128	256	512	1024	2048	4096	8192	16384
2	26003	16791	7172	4744	3373	3061	2994	2994
4	48254	21656	13250	5673	3632	2803	2576	2543
8	169540	44904	19141	10928	4745	2950	2398	2214
16	279236	165115	39728	15904	7876	3828	2380	2011
32	332851	270086	153176	32282	13230	5974	2961	1950
64	376284	284027	234688	115886	21930	9963	4328	2331

LRU	PAGINAS ESCRITAS							
	128	256	512	1024	2048	4096	8192	16384
2	5162	3420	1859	1260	892	468	0	0
4	8668	4511	2939	1539	1009	693	253	0
8	44252	8084	3924	2516	1348	848	530	73
16	52232	43824	7194	3342	1982	1098	689	401
32	53682	51969	42709	5523	2809	1570	822	536
64	57154	53250	51597	34924	3361	2125	1195	653

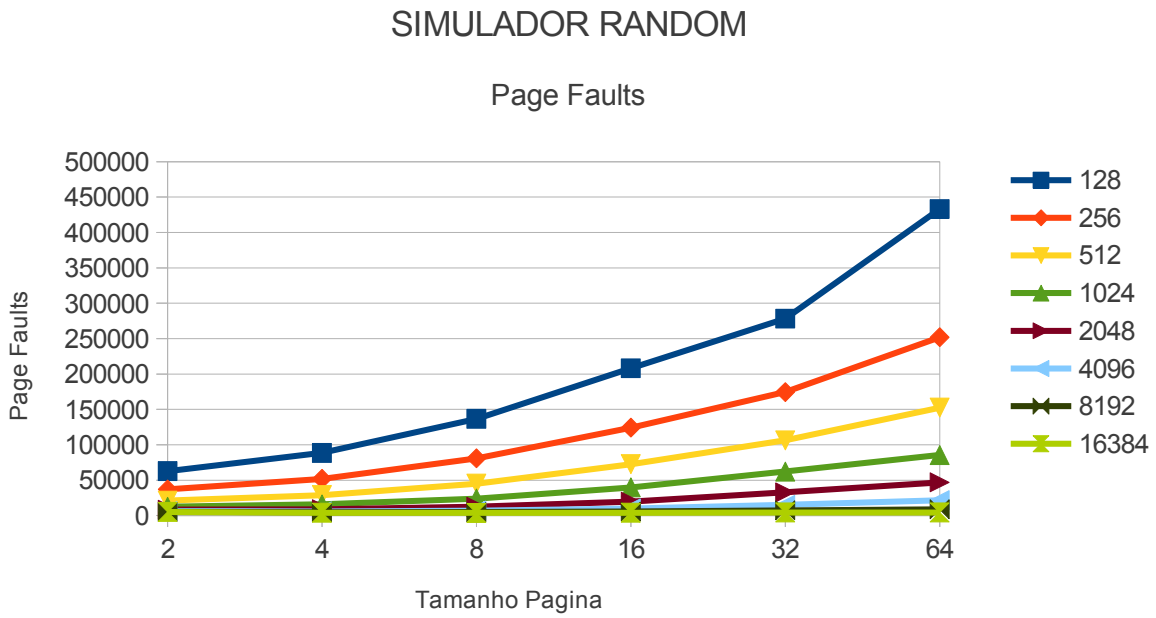
RANDOM
compilador.log

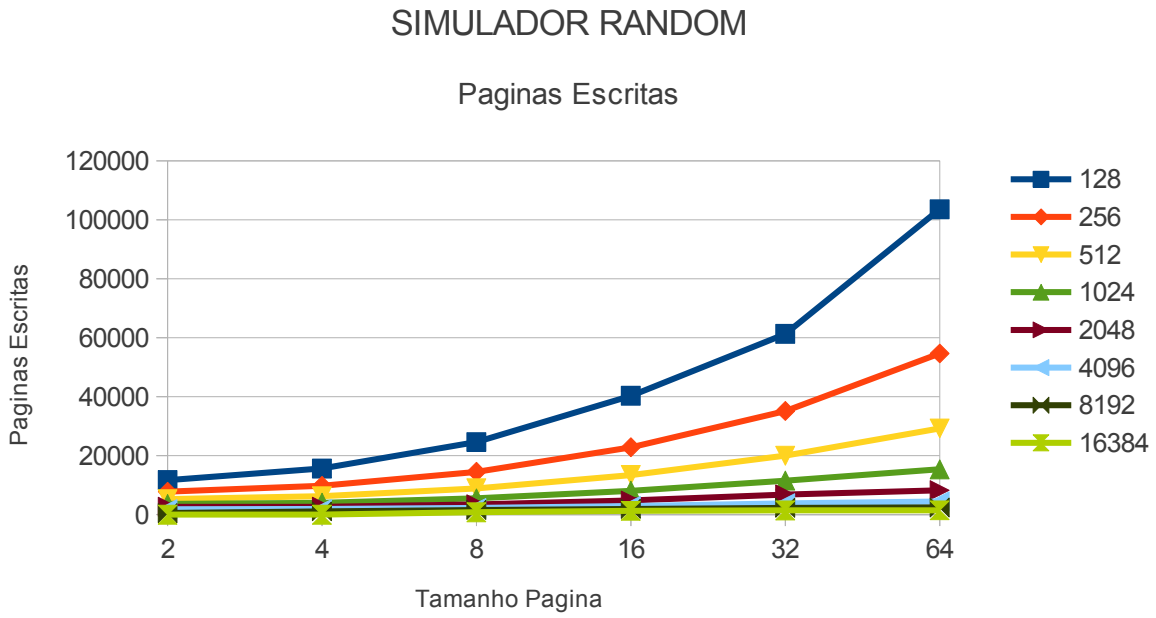




COMPILADOR RANDOM	PAGE FAUL							
	128	256	512	1024	2048	4096	8192	16384
2	91118	63697	41368	22624	10871	5182	3621	3621
4	106522	75116	51496	32022	16291	7151	3422	2852
8	144007	100470	69677	46247	27178	12605	5123	2529
16	200908	134197	91899	61856	39198	20817	8676	3441
32	276743	181763	119875	79584	51441	30893	14673	5545
64	378345	239465	156045	102135	66083	41335	23629	10411
RANDOM	PAGINAS ESCRITAS							
	128	256	512	1024	2048	4096	8192	16384
2	9581	6593	4316	2514	1385	641	0	0
4	11820	8001	5374	3452	1916	944	289	0
8	17433	11438	7644	5028	3073	1593	708	108
16	26550	16498	10634	7028	4354	2412	1159	463
32	39915	24352	14876	9236	5606	3312	1689	743
64	61213	36911	22158	13170	7887	4665	2566	1157

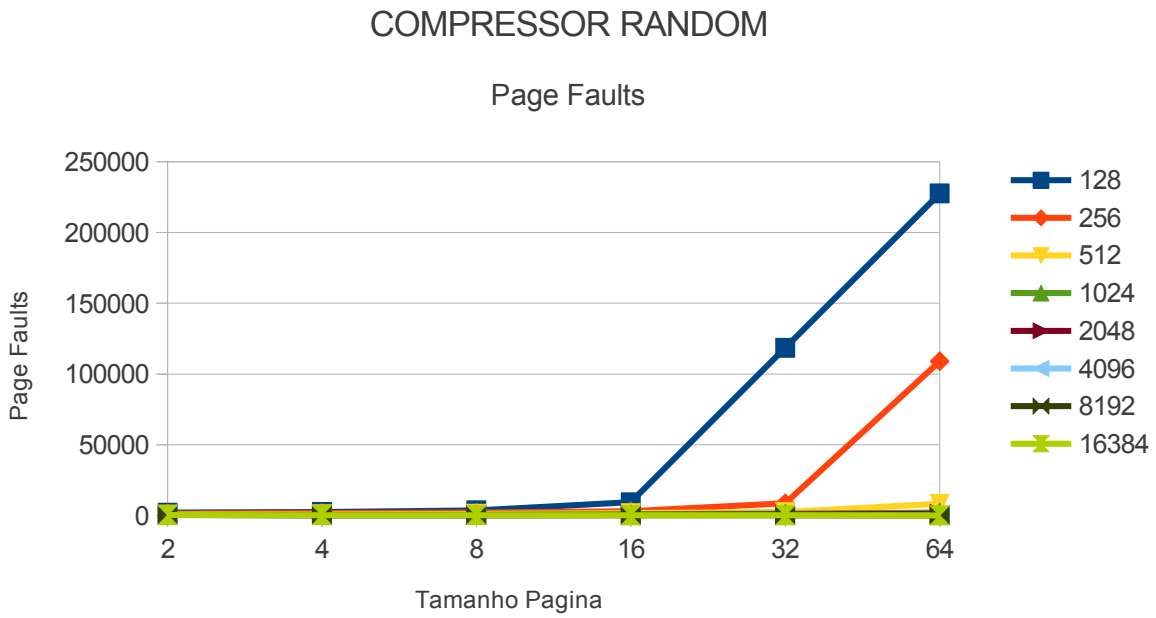
simulador.log

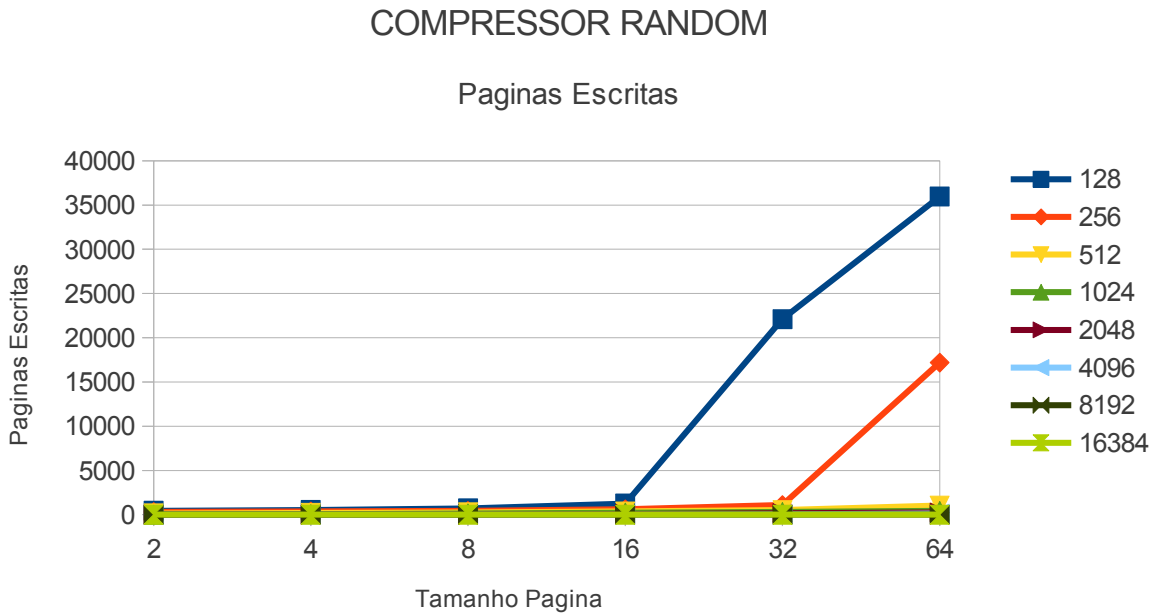




SIMULADOR RANDOM	PAGE FAUL							
	128	256	512	1024	2048	4096	8192	16384
2	62797	36907	21264	12433	8318	6344	5307	5144
4	88596	51830	28950	15779	8952	5913	4515	3890
8	136696	80917	45065	23931	12482	7019	4844	3756
16	208203	124207	72655	39849	19729	9995	5711	3960
32	278326	174522	106342	62165	32736	15131	7354	4302
64	433261	252032	152415	85620	46913	21676	8851	4337
RANDOM	PAGINAS ESCRITAS							
	128	256	512	1024	2048	4096	8192	16384
2	11701	7780	5249	3593	2650	1871	525	0
4	15645	9794	6276	4034	2674	1901	1124	0
8	24636	14524	8909	5494	3496	2255	1599	815
16	40318	22848	13458	8121	4781	2949	1926	1300
32	61280	35149	19997	11537	6834	3848	2313	1461
64	103603	54654	29253	15375	8297	4511	2423	1465

compressor.log





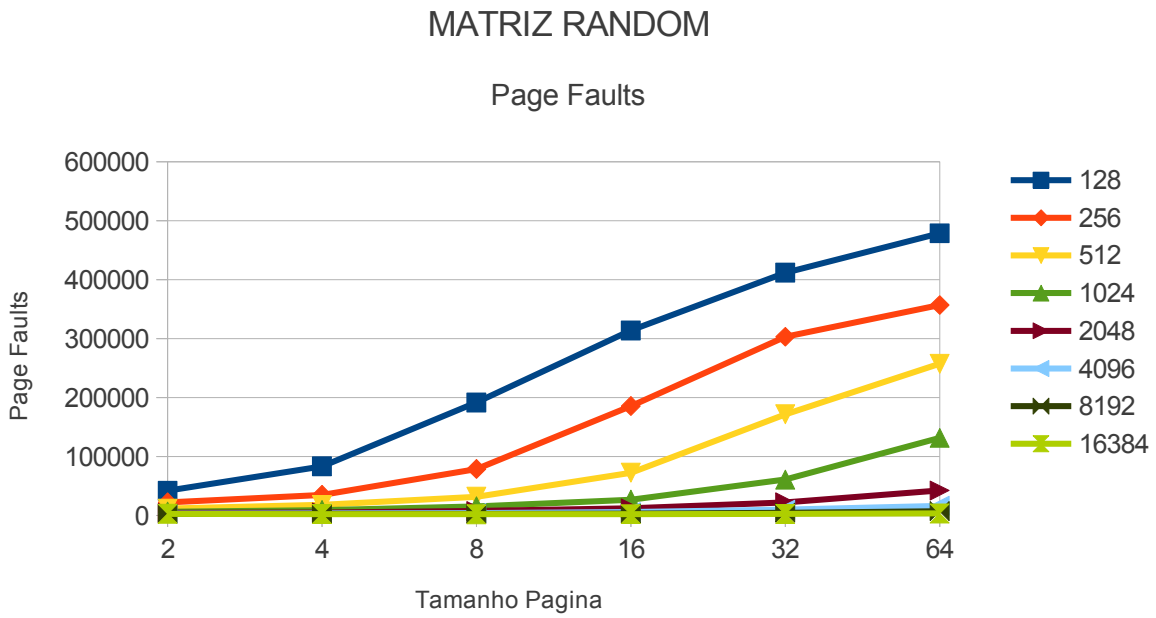
COMPRESSOR
RANDOM PAGE FAUL

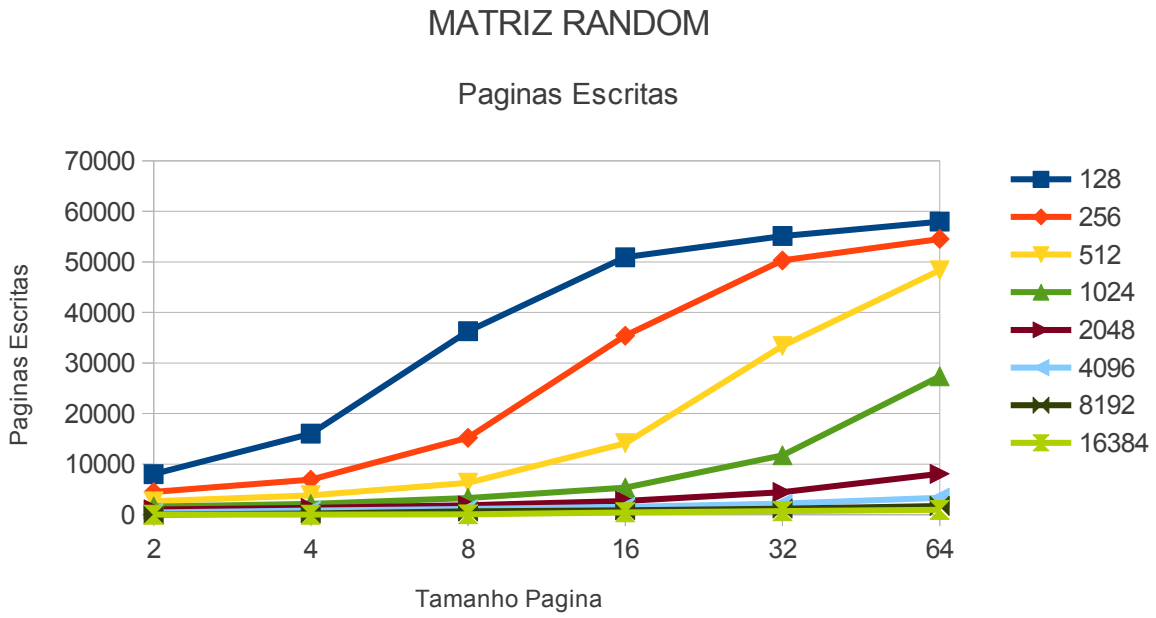
	128	256	512	1024	2048	4096	8192	16384
2	2130	1390	742	419	419	419	419	419
4	2588	1674	899	399	317	317	317	317
8	3706	2055	1197	661	255	255	255	255
16	9411	3068	1719	975	457	209	209	209
32	118509	8701	2586	1328	709	268	172	172
64	227610	109066	8294	2167	1090	475	138	137

RANDOM PAGINAS ESCRITAS

	128	256	512	1024	2048	4096	8192	16384
2	443	307	131	0	0	0	0	0
4	529	363	195	48	0	0	0	0
8	717	434	253	134	0	0	0	0
16	1251	614	372	224	83	0	0	0
32	22110	1087	514	294	163	46	0	0
64	35972	17194	1022	435	234	116	4	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	8031	4484	2652	1618	1038	449	0	0
4	16028	6908	3831	2112	1305	802	250	0
8	36285	15204	6302	3295	1846	1113	636	65
16	50917	35422	14127	5370	2717	1517	868	450
32	55128	50289	33371	11745	4470	2170	1182	684
64	57980	54531	48356	27367	8067	3348	1714	949

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.