Randyll Bearer: RLB97: CS1550: Project 3: VMSIM

Recorded Statistics:

OPT(): Not Completed

RAND():

Frames	8	16	32	64
BZIP-FAULTS	44209	4463	2583	1575
BZIP-WRITES	16488	1478	873	539
GCC-FAULTS	217060	150613	106286	75031
GCC-WRITES	37370	24832	17213	12130

CLOCK():

Frames	8	16	32	64
BZIP-FAULTS	34926	3418	2120	1301
BZIP-WRITES	12155	1094	700	423
GCC-FAULTS	168291	115520	83860	57953
GCC-WRITES	22848	14349	11522	8685

NRU() 100 Refresh Rate:

Frames	8	16	32	64
BZIP - FAULTS	34223	6274	5122	3113
BZIP - WRITES	9549	1252	1346	513
GCC - FAULTS	235415	132982	95720	92655
GCC - WRITES	17651	13611	9170	8444

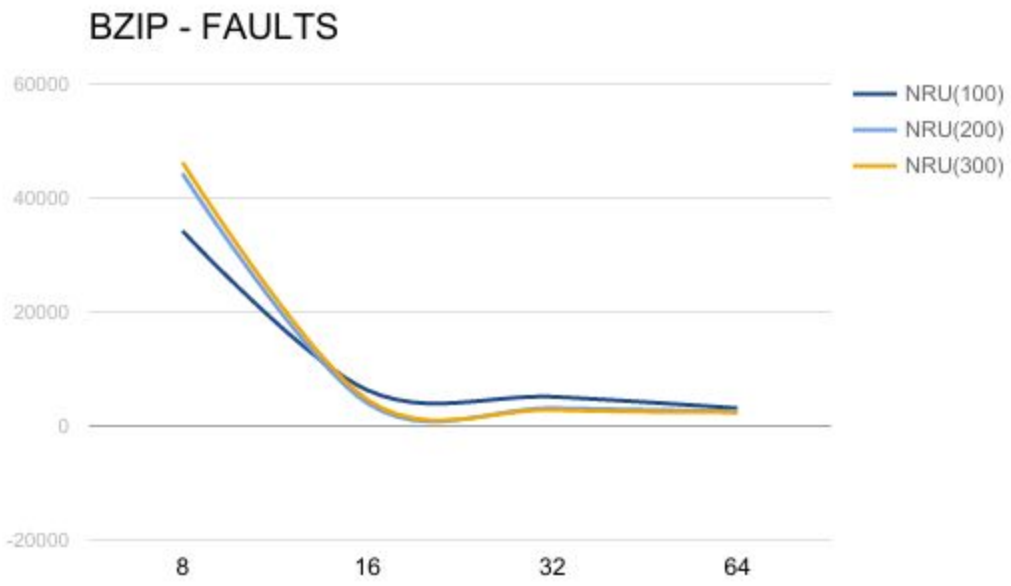
NRU() 200 Refresh Rate:

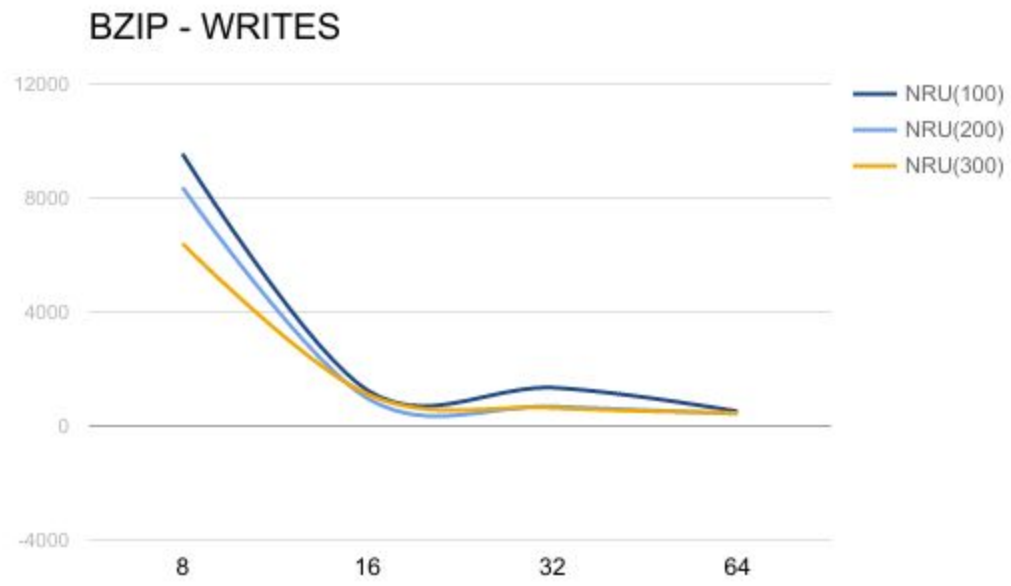
Frames	8	16	32	64
BZIP - FAULTS	44286	3967	3145	2573
BZIP - WRITES	8371	969	686	433
GCC - FAULTS	271318	174691	90702	74199
GCC - WRITES	16811	13819	10520	6928

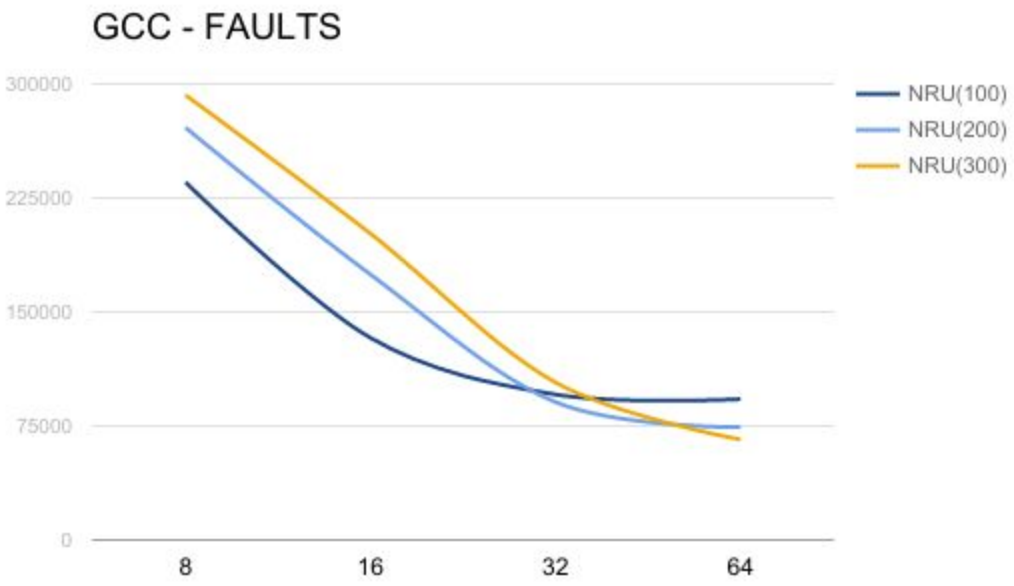
NRU() 300 Refresh Rate:

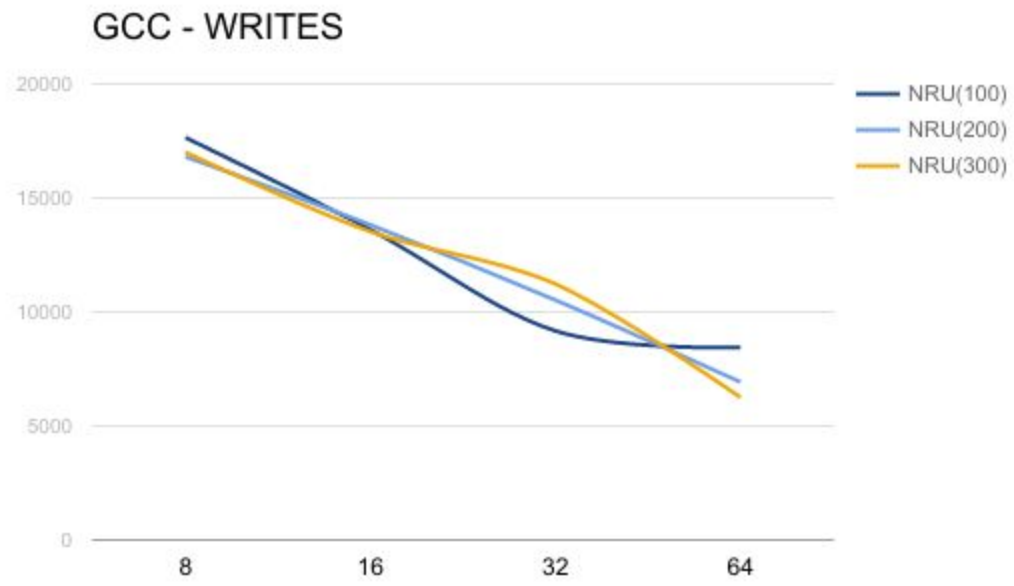
Frames	8	16	32	64
BZIP - FAULTS	46273	4608	2826	2400
BZIP - WRITES	6401	1112	636	459
GCC - FAULTS	292709	201555	103667	66114
GCC - WRITES	17008	13523	11225	6253

NRU REFRESH RATE RESULTS





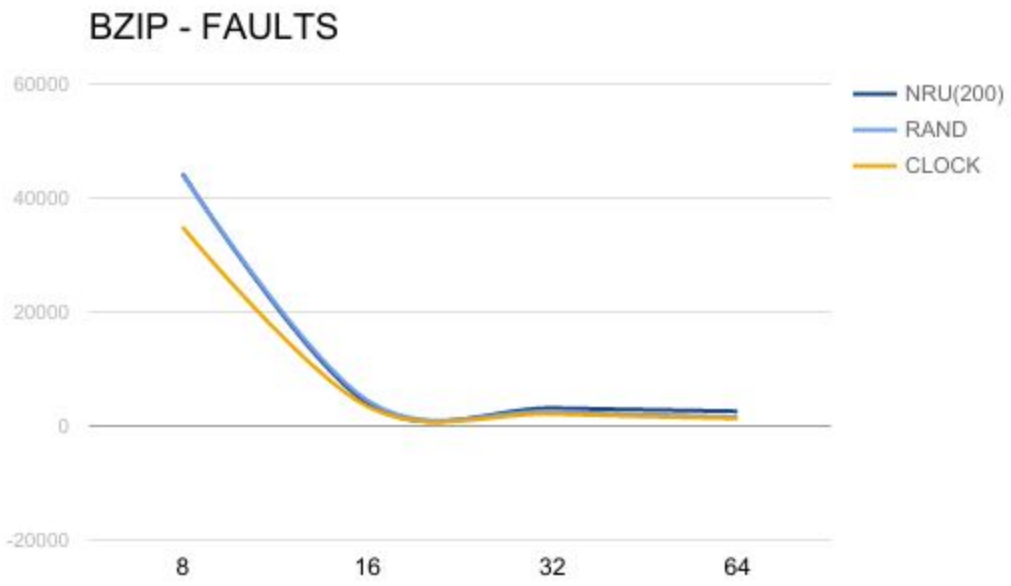


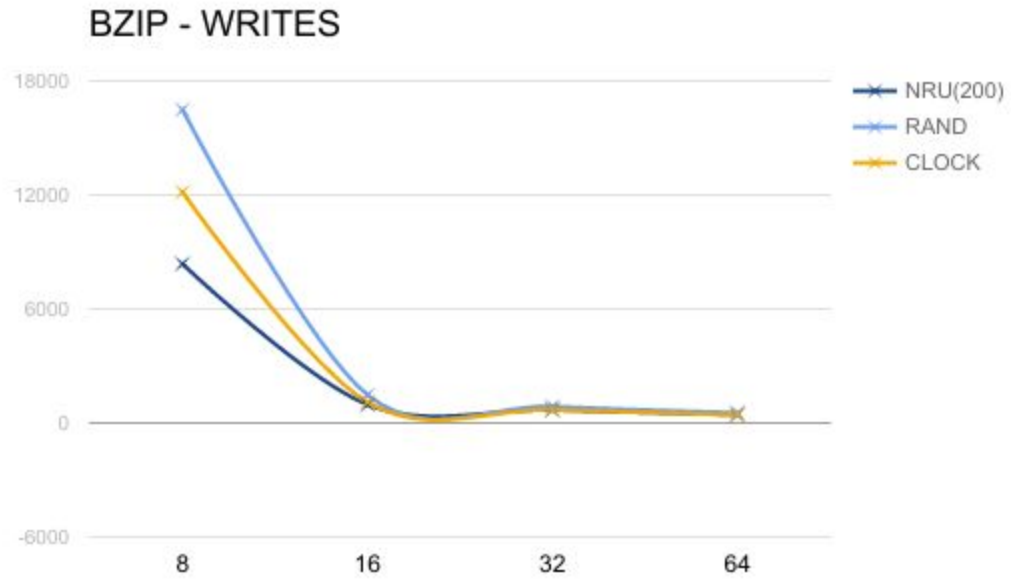


NRU CONCLUSION:

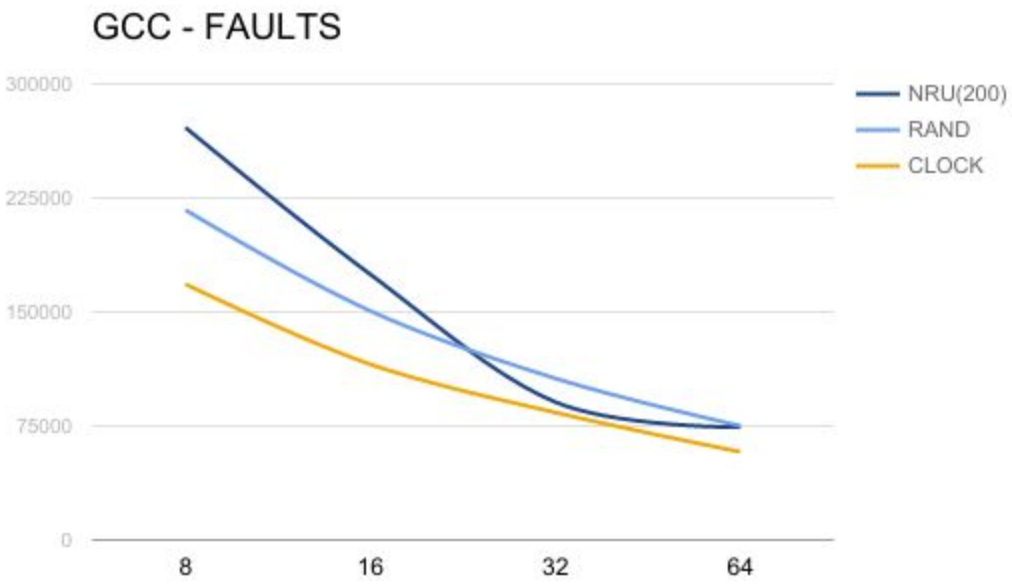
- Naturally which refresh rate I used in my OS would depend on the amount of frames I would expect to have, but I think 200 strikes a good balance out of the data I have collected. Going forward in the graphs, the NRU with refresh rate 200 will be used to represent the NRU Page Replacement Algorithm.

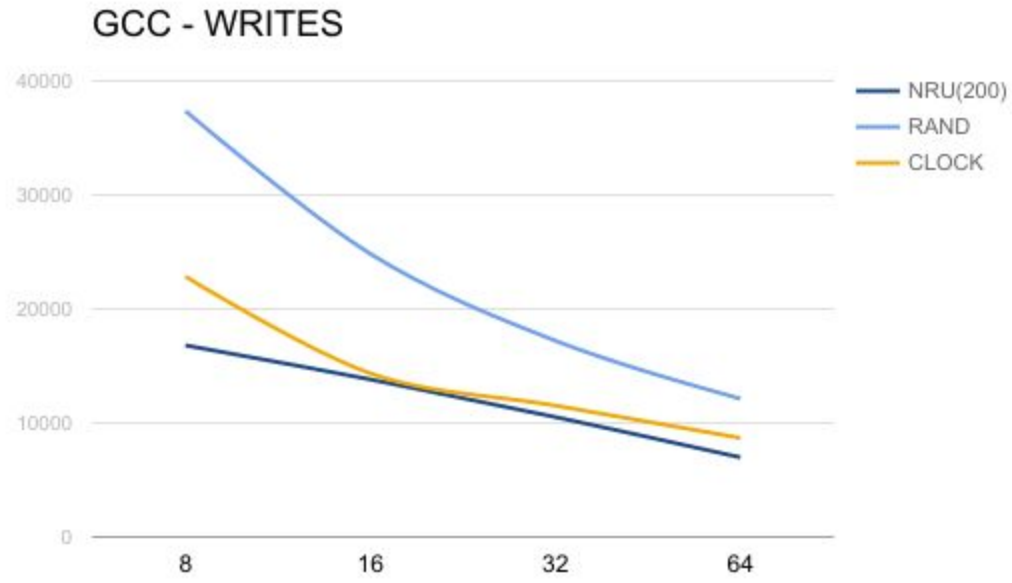
BZIP: ALL ALGORITHMS





GCC: ALL ALGORITHMS





CONCLUSION ON PAGE REPLACEMENT ALGORITHMS:

- I think the results show that the Clock algorithm would be the most apt for use in an OS. Plus, it requires a bit less overhead than the nearest competitor NRU. RAND always has the chance of literally being OPT, but on average it’s just not going to be worth it so Clock it is.