Emilio Quiambao

Homework 6

Overview

This program simulates FIFO and LRU page replacement algorithms. It accepts four command line arguments (i.e. page size, memory size, page replacement algorithm, and the number of frames allocated, respectively) After reading the command line arguments, the program reads hex numbers which represent the virtual addresses of memory references. This can either be manually inputted by the user or indicated by the start while inputting the command line arguments by referring to a text file in the same directory.

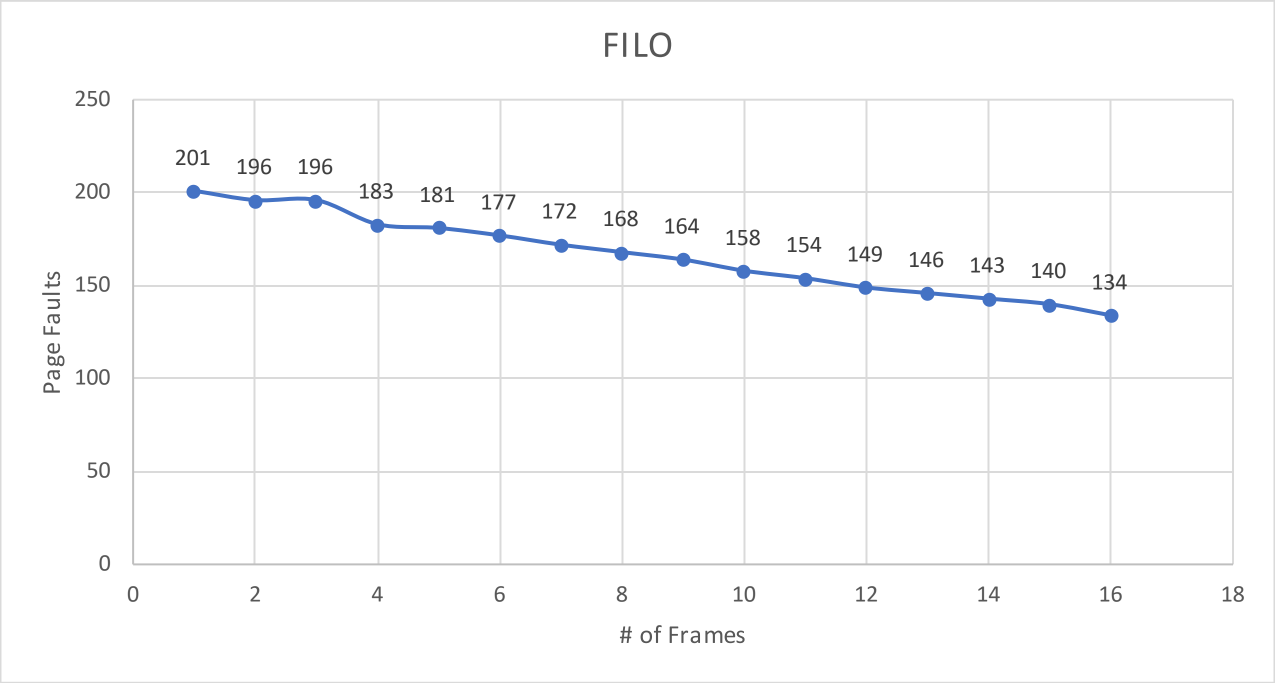
The FIFO algorithm simply loops and checks if the current page reference is in the array. If it is, then it stays the same and does not add to the page fault counter. However, if it is, then it replaces the oldest page in the memory.

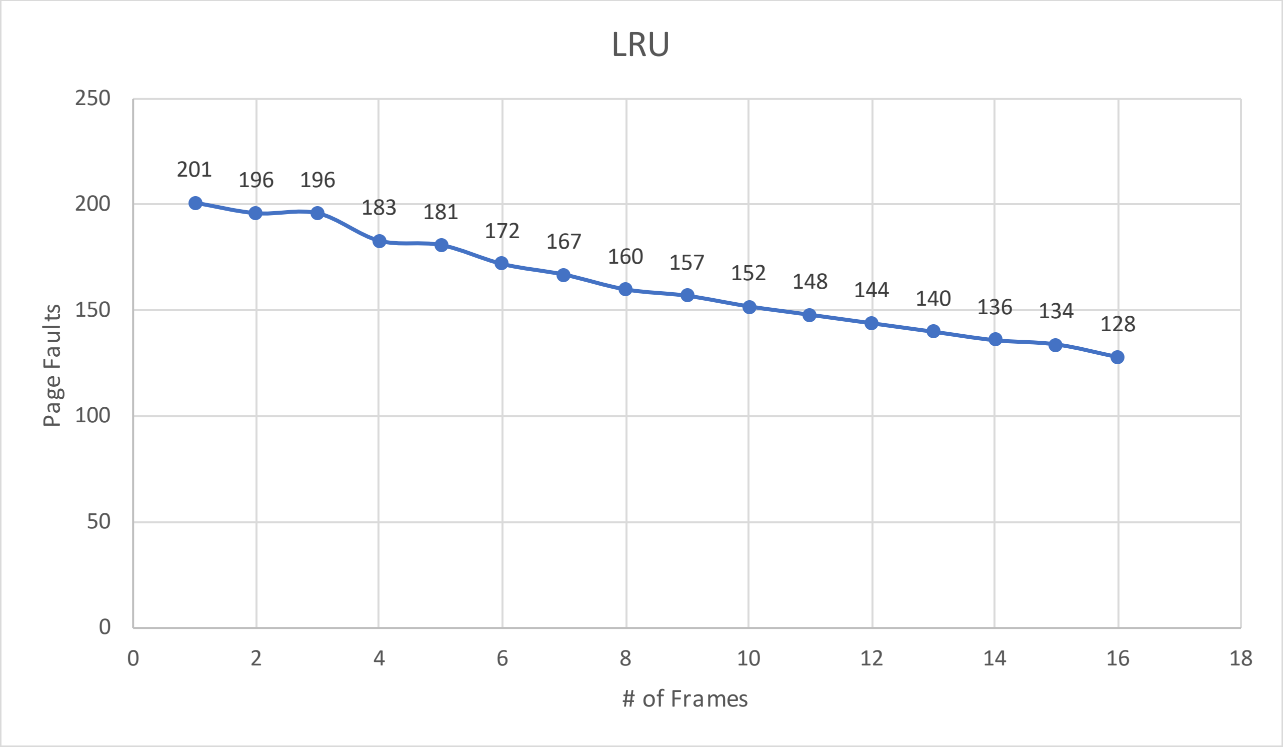
The LRU algorithm involves several tables. The two extra tables checkTable A and B keeps track of the least recently used page. The algorithm first checks if the current page is in the frames array. If it is, we continue looping. If it isn't, we place it in "innerIndex", replace it with the least recently used page and determine the next least recently used page.

I created a text file named test.txt containing 100 random memory references. Below is the data using 4KB page size and 64KB memory size running tests from 1 – 16 allocated frames. Belady's anomaly did not occur during either test since the page faults consistently went down as the page frames increase. I have found that FILO and LRU had similar page faults in the beginning, but LRU started to have a 5~8 less page fault difference after the fifth frame iteration.

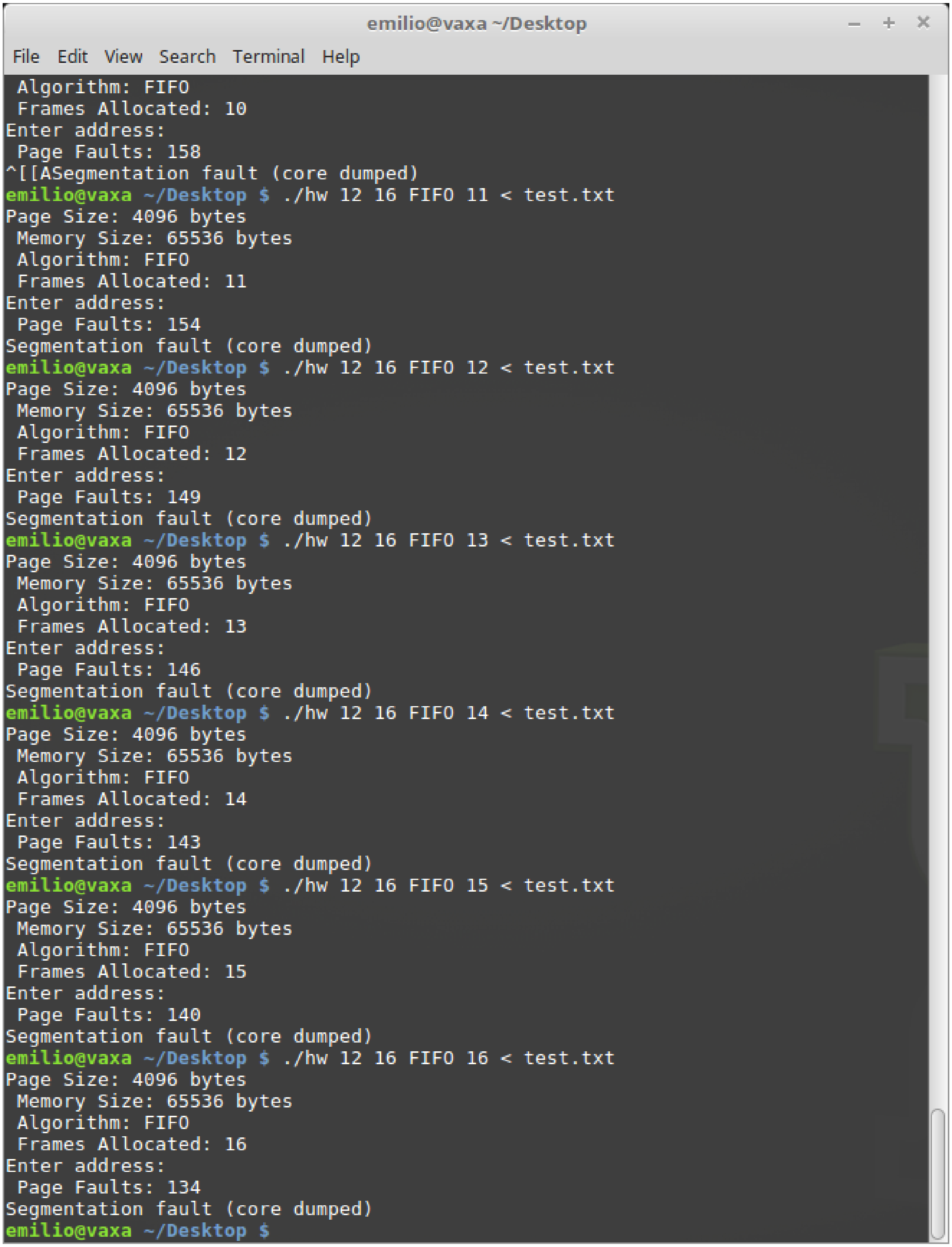
Data and Graphs

FILO		LRU	
Frames	Faults	Frames	Faults
1	201	1	201
2	196	2	196
3	196	3	196
4	183	4	183
5	181	5	181
6	177	6	172
7	172	7	167
8	168	8	160
9	164	9	157
10	158	10	152
11	154	11	148
12	149	12	144
13	146	13	140
14	143	14	136
15	140	15	134
16	134	16	128

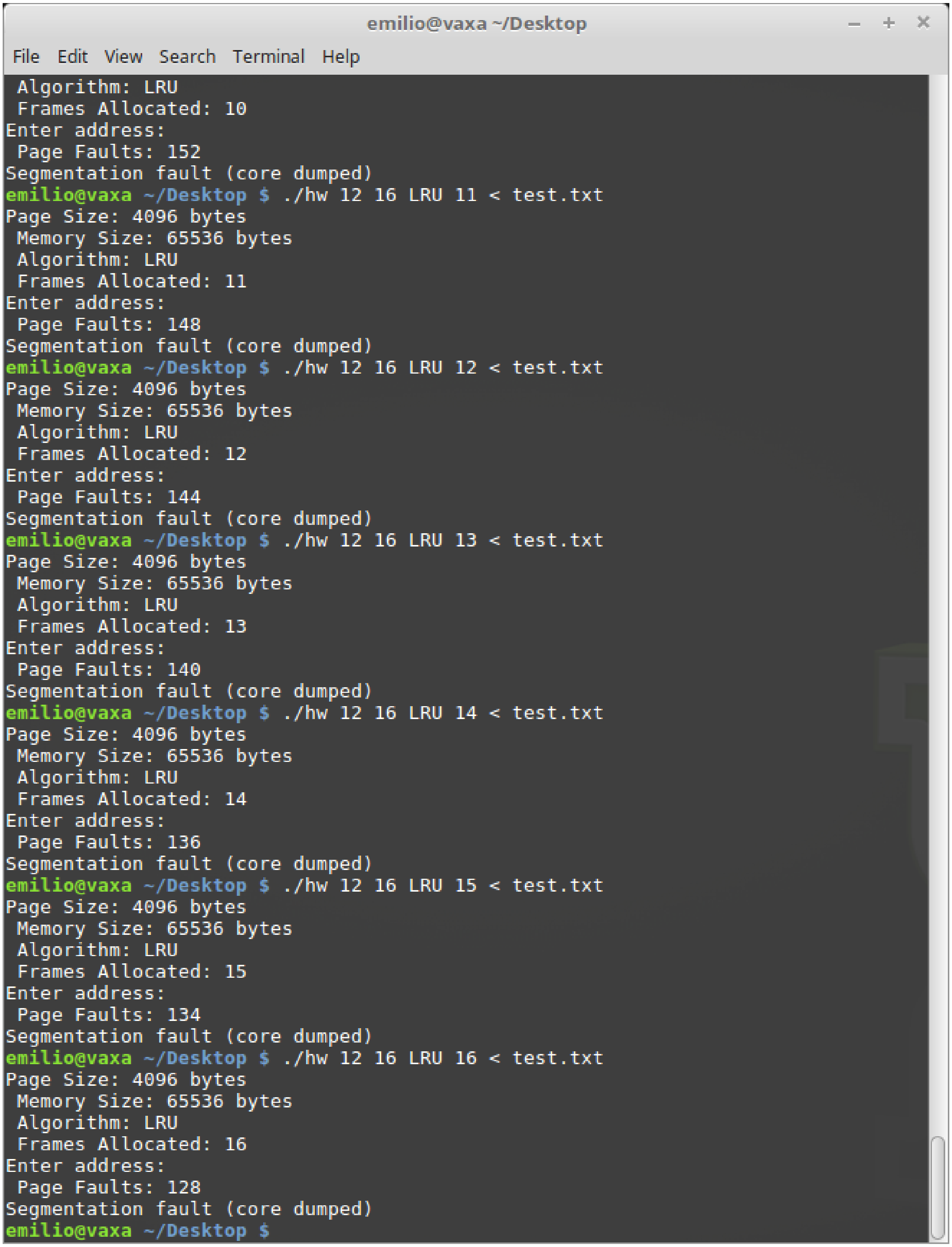

FILO
250
200
150
100
50
0
Page Faults
201
196
196
183
181
177
172
168
164
158
154
149
146
143
140
134
0
2
4
6
8
10
12
14
16
18
# of Frames


LRU
250
200
150
100
50
0
Page Faults
201
196
196
183
181
172
167
160
157
152
148
144
140
136
134
128
0
2
4
6
8
10
12
14
16
18
# of Frames

Screenshots



```
 Algorithm: FIFO
 Frames Allocated: 10
Enter address:
 Page Faults: 158
^[[ASegmentation fault (core dumped)
emilio@vaxa ~/Desktop $ ./hw 12 16 FIFO 11 < test.txt
Page Size: 4096 bytes
 Memory Size: 65536 bytes
 Algorithm: FIFO
 Frames Allocated: 11
Enter address:
 Page Faults: 154
Segmentation fault (core dumped)
emilio@vaxa ~/Desktop $ ./hw 12 16 FIFO 12 < test.txt
Page Size: 4096 bytes
 Memory Size: 65536 bytes
 Algorithm: FIFO
 Frames Allocated: 12
Enter address:
 Page Faults: 149
Segmentation fault (core dumped)
emilio@vaxa ~/Desktop $ ./hw 12 16 FIFO 13 < test.txt
Page Size: 4096 bytes
 Memory Size: 65536 bytes
 Algorithm: FIFO
 Frames Allocated: 13
Enter address:
 Page Faults: 146
Segmentation fault (core dumped)
emilio@vaxa ~/Desktop $ ./hw 12 16 FIFO 14 < test.txt
Page Size: 4096 bytes
 Memory Size: 65536 bytes
 Algorithm: FIFO
 Frames Allocated: 14
Enter address:
 Page Faults: 143
Segmentation fault (core dumped)
emilio@vaxa ~/Desktop $ ./hw 12 16 FIFO 15 < test.txt
Page Size: 4096 bytes
 Memory Size: 65536 bytes
 Algorithm: FIFO
 Frames Allocated: 15
Enter address:
 Page Faults: 140
Segmentation fault (core dumped)
emilio@vaxa ~/Desktop $ ./hw 12 16 FIFO 16 < test.txt
Page Size: 4096 bytes
 Memory Size: 65536 bytes
 Algorithm: FIFO
 Frames Allocated: 16
Enter address:
 Page Faults: 134
Segmentation fault (core dumped)
emilio@vaxa ~/Desktop $
```



```
 Algorithm: LRU
 Frames Allocated: 10
Enter address:
 Page Faults: 152
Segmentation fault (core dumped)
emilio@vaxa ~/Desktop $ ./hw 12 16 LRU 11 < test.txt
Page Size: 4096 bytes
 Memory Size: 65536 bytes
 Algorithm: LRU
 Frames Allocated: 11
Enter address:
 Page Faults: 148
Segmentation fault (core dumped)
emilio@vaxa ~/Desktop $ ./hw 12 16 LRU 12 < test.txt
Page Size: 4096 bytes
 Memory Size: 65536 bytes
 Algorithm: LRU
 Frames Allocated: 12
Enter address:
 Page Faults: 144
Segmentation fault (core dumped)
emilio@vaxa ~/Desktop $ ./hw 12 16 LRU 13 < test.txt
Page Size: 4096 bytes
 Memory Size: 65536 bytes
 Algorithm: LRU
 Frames Allocated: 13
Enter address:
 Page Faults: 140
Segmentation fault (core dumped)
emilio@vaxa ~/Desktop $ ./hw 12 16 LRU 14 < test.txt
Page Size: 4096 bytes
 Memory Size: 65536 bytes
 Algorithm: LRU
 Frames Allocated: 14
Enter address:
 Page Faults: 136
Segmentation fault (core dumped)
emilio@vaxa ~/Desktop $ ./hw 12 16 LRU 15 < test.txt
Page Size: 4096 bytes
 Memory Size: 65536 bytes
 Algorithm: LRU
 Frames Allocated: 15
Enter address:
 Page Faults: 134
Segmentation fault (core dumped)
emilio@vaxa ~/Desktop $ ./hw 12 16 LRU 16 < test.txt
Page Size: 4096 bytes
 Memory Size: 65536 bytes
 Algorithm: LRU
 Frames Allocated: 16
Enter address:
 Page Faults: 128
Segmentation fault (core dumped)
emilio@vaxa ~/Desktop $
```