330 Final Project - Inter/Summer 2016

William Briguglio

I confirm that I will keep the content of this project confidential. I confirm that I have not received any unauthorized assistance in preparing for or writing this project. I acknowledge that a mark of 0 may be assigned for copied/plagiarized work. William Briguglio 104 205 372.

Sample Run

My program is saved as "FP.c" I compiled and ran this code with **`cc FP.c`** and **`./a.out addresses.txt`** . The input to this run was the provided addresses.txt file and BACKING_STORE.bin file. The output it produces was saved in a file called "results.txt" as well as printed to the console. The input/output of this run as well as the source code can be found in the zip file I submitted. As shown below, my program produced 244 page faults for 1000 addresses, giving a page fault rate of 0.244.

```
Vitual address: 2301    Physical addrss: 35325  Value: 0
Vitual address: 7736    Physical addrss: 57912  Value: 0
Vitual address: 31260   Physical addrss: 23324  Value: 0
Vitual address: 17071   Physical addrss: 175    Value: -85
Vitual address: 8940    Physical addrss: 46572  Value: 0
Vitual address: 9929    Physical addrss: 44745  Value: 0
Vitual address: 45563   Physical addrss: 46075  Value: 126
Vitual address: 12107   Physical addrss: 2635   Value: -46
Number of Translated Addresses = 1000
Page Faults = 244
Page Fault Rate = 0.244
briguglw@charlie:~/330/FP$
```

Implementation

Physical memory is implemented as an array of type unsigned char and size 65536. This is because unsigned char is worth 1 byte so this would give me a 65536 byte physical memory

The Page Table is implemented as a 256 by 2 array of type unsigned long. The first column is the corrisponding frame and the second is the valid/invalid bit. The page table is initialized to 256 for all frames (since this is larger than any possible frame number) and invalid for all entries

After scanning in the logical address it is passed to a function called findPhysAddr.
In this funciton:

1)the logical address is broken up into its page and offset

2)a)if the page table's valid/invalid bit is set to **invalid** at pageTable[page] then

-the physical address is found by concatenating the number of entries in the page table to the front of the offset.

-the number of entries in the page table is incremented by 1

-the number of page faults is incremented by 1

-the frame number is extracted from this physical address

-pageTable[page] is set to the frame number

-this frame is copied from the backing store into physical memory

-the value is found in physical memory using the frame number and offset

b)if the page table's valid/invalid bit is set to **valid** at pageTable[page] then

-the frame is copied from pageTable[page]

-the physical address is found by appending the offset to this frame

-the value is found in physical memory using the frame number and offset

3)The Logical and Physical address and value are then printed