CS 346 – Lab #8 – Simple page table simulation

Today you will write a program that does some nuts and bolts calculations for page table. The purpose of a page table is to give us the correspondence between a logical address and a physical address. It specifically can tell us if a datum resides in RAM or not.

To begin, please log in and create a directory called lab08. Copy all of the files from my directory ~chealy/www/cs346/lab08. You will find an unfinished Driver.java program as well as three example input files called 1.txt, 2.txt and 3.txt. Please look at an input file so you understand its information.

All you need to do is add the necessary code to Driver.java. The comments in the program should guide your work. In particular, the comment at the top of the program lists 9 steps that you are to complete.

- For a logical address, determine the number of bits allocated to its page number and offset
- For a physical address, determine the number of bits allocated to its page number and offset
- Determine the number of logical pages
- Determine the number of physical pages
- Create a page table (array) based on the information you just calculated. I have already given you a statement that will initialize one of the entries of the page table.
- Write a loop that will read logical address from the input file. For each address, figure out its logical page number and look it up in the page table. Is it a hit or a miss? If it's a miss, update the page table, assuming that the next sequentially numbered physical page is available.
- Detect if any data address that you read maps to physical page 0. If so, announce this as a "segmentation fault."
- Detect if we run out of physical pages. If so, we have to stop accepting addresses.
- When your program is done with the input, print out the contents of the page table, using the function I wrote for you.

As you work, please take note of the following:

First, your program will do some routine calculations related to the sizes of logical and physical memory. We do this so that we know how to partition an address, and so that we know how many logical and physical pages we have. Anytime you see a statement that assigns the value –123 to a variable, this represents a place where you need to do a useful calculation with the appropriate formula.

You might find the functions at the bottom of the program useful in your work. Namely, log2(), findLogicalPageNumber() and printPageTable(). Read the documentation to see how to call these functions.

Show me the output you get when you run your program with the 3 input files. To help you check your program, here is the output you should get if you run with input 1.txt and 2.txt:

```
Size of RAM = 256 MB.
Size of a page = 4 KB.
We assume the logical address has 32 bits.
```

```
The page offset has 12 bits.
The logical page number has 20 bits.
The physical address has 28 bits.
The physical page number has 16 bits.
There are 1048576 logical pages.
There are 65536 physical pages.
Program text has logical page number 1024.
Address 0x10010000 has logical page number 65552.
  page table miss
Address 0x10010004 has logical page number 65552.
  page table hit
Address 0x10010008 has logical page number 65552.
  page table hit
Address 0x12345678 has logical page number 74565.
  page table miss
Address 0x00400118 has logical page number 1024.
  Segmentation Fault.  Continuing...
We're done.
Page table:
  logical page 1024 mapped to physical page 0
  logical page 65552 mapped to physical page 1
  logical page 74565 mapped to physical page 2

Size of RAM = 128 MB.
Size of a page = 1 KB.
We assume the logical address has 32 bits.
The page offset has 10 bits.
The logical page number has 22 bits.
The physical address has 27 bits.
The physical page number has 17 bits.
There are 4194304 logical pages.
There are 131072 physical pages.
Program text has logical page number 4096.
Address 0x10010000 has logical page number 262208.
  page table miss
Address 0x10010004 has logical page number 262208.
  page table hit
Address 0x10010008 has logical page number 262208.
  page table hit
Address 0x12345678 has logical page number 298261.
  page table miss
Address 0x00400118 has logical page number 4096.
  Segmentation Fault.  Continuing...
We're done.
Page table:
  logical page 4096 mapped to physical page 0
  logical page 262208 mapped to physical page 1
  logical page 298261 mapped to physical page 2
```