ICS 53, Winter 2018

Lab 3: Virtual Memory Simulator

You will implement a VM Simulator which will simulate the operation of a virtual memory system. The system will accept commands to read/write from/to a virtual address space. Your system will need to correctly move pages between disk and main memory in order to satisfy access requests. Your system will need to properly maintain the page table as part of this process.

Parameters of the Virtual Memory System

Each address in memory contains a single integer. All memory locations are initialized to the value of -1 but only positive integers can be written to memory locations by the user. Your virtual memory system will have 16 addresses (0 - 15) and main memory has 8 addresses (0 - 7). Each page contains 2 addresses, so your virtual memory system will have 8 pages and your main memory will have 4 pages. Pages are numbered sequentially starting at the lowest memory address. So page 0 contains addresses 0 and 1, page 1 contains addresses 2 and 3, and so on.

User Interface of VM Simulator

The user will type in a sequence of commands and your program will perform the operation specified by each command. Your program will execute in a loop which starts with your program printing a '$' prompt at the beginning of the line in order to let the user know that your program is ready to receive a new command. The user will type in a command, followed by any necessary arguments, and then hit the 'enter' key to indicate that the command is complete. Your program will then execute the command, printing data to the screen is necessary, print a '$' prompt on a new line to receive a new command.

Commands

Your program must process the following user commands.

1. **read <address>**: This command prints the contents of a memory address. The command takes one argument which is the virtual address of the memory location to be read.
2. **write <address> <num>**: The command writes data to a memory location. The command takes two arguments, the virtual address to write the data to, and a positive integer which will be written to the address.
3. **showmain <ppn>**: This command prints the contents of a physical page in main memory. The command takes one argument which is the number of the physical page to be printed. Each page contains two addresses, so the contents of both addresses should be printed, together with their associated addresses in main memory. You can see the format from the example below which shows the

contents of physical page 1 which includes the value 100 at address 2, and the value 101 at address 3.

```
Address    Contents
2          100
3          101
```

4. **showdisk <dpn>**: This command prints the contents of a virtual page on disk. The command takes one argument which is the number of the disk page to be printed. page frame contains two addresses, so the contents of both addresses should be printed, together with their associated addresses in main memory. The format for printing the contents of the disk page is the same as the format for the showmain command.
5. **showptable**: This command prints the contents of the page table. Your virtual memory system contains 8 virtual pages, so this command will print 8 page table entries. Each page table entry contains three fields of information about a page.
    - *Valid bit*: 1 indicates that the corresponding page is in main memory and 0 indicates that the page is on disk.
    - *Dirty bit*: 1 indicates that the corresponding page has been written to while in main memory. 0 indicates that the page has not been written to since it has been in main memory. The dirty bit has no meaning if the page is not in main memory.
    - *Page Number (PN)*: This is an integer indicating the number of the page (in main memory or disk) where the virtual page can be found in memory. The PN refers to a main memory physical page when the Valid bit is equal to 1, and the PN refers to a disk page when the Valid bit is 0.

    The showptable command will print the contents of each page table entry on a separate line in a tabular form with four columns: page number, valid bit, dirty bit, and PFN. Below is an example of three page table entries of the output of the showptable command.

```
VPageNum  Valid     Dirty     PN
0         1         0         0
1         1         1         3
2         0         0         2
```

    Although this example only shows 3 page table entries for brevity, your showptable command should print all 8 page table entries.
6. **quit**: This command quits the program.

Handling page faults

When a page fault occurs, a disk page must be copied into main memory. If there is one or more page in main memory which is available (there is not virtual page mapped to it) then the disk page should be copied into the available main memory page with the lowest page number. If all pages in main memory are in use then a victim page must be chosen for eviction from main memory. Choose the victim page which has been

accessed least recently. Remember that the victim page must be copied back to disk before it is evicted, if its dirty bit is 1.

When a victim page is copied back to disk, always copy it back to the disk page whose number is the same as the number of the virtual page.

Initial Conditions

Assume that all virtual pages are initially on disk, so the valid bits of all page table entries are equal to 0. Assume that the disk page number of each virtual page is the same as the virtual page number.

Additional Note:

For the sake of simplicity, the page table will be assumed to be stored separately from main memory. This means that when you issue the showmain command you will not see the page table contents. Also, when you issue the showptable command you will not see main memory.

Submission Instructions:

There will be a dropbox on Canvas which you will use to submit your code. You should submit a single C source code file. The first line of your submitted file must be a comment which includes the name and ID number of you and your partner (if you are working with a partner). Each student should upload a source code file, even if you are working with a partner. When working with a partner, the source code files that you upload must be identical. The code must compile and execute on the openlab machines. The name of your C source code file should be “Lab3.c”.