COMPUTER SYSTEMS
PROJECT 5
Implenting page swapping

1 Looking inside `vmsim`

For this project, you will be working within `vmsim`. I have provided all of its code (modified somewhat from the previous project), including my own MMU implementation, as a starting point.

Grabbing new source: To get started, create a directory for this project and then, within it, get the source code:

```
$ wget -nv -i https://bit.ly/COSC-171-project-5-source
```

The most immediate change is the presence of the `bs.c` and `bs.h` files, which implement a simulated *backing store*—a disk-like larger storage that allows you to read and write whole blocks (each conveniently 4 KB).

Additionally, you can now look inside `vmsim.c` to see how it works. Of paritcular interest is the function `vmsim_map_fault()`, since it is responsible for handling MMU translations that fail. You should also notice, in `mmu.c`, that the MMU now does two new, important things:

1. **Test the *resident* bit:** Each page table entry uses the bit in position 0 to indicate whether that simulated page is mapped to an honest to goodness *real* page that is available and ready for use. If this bit is 0, the translation fails.

2. **Set the *referenced* and *dirty* bits:** When a translation succeeds, the bit at position 1 is set (to 1), indicated that this simulated page has been referenced. If the reference is a *write* operation, then the bit at position 2 is set (to 1), marking the simulated page as *dirty*.[1]

There are likely other features that you will want to take in, including a number of `#define` macros that I've used for manipulating bits, various helpful constants, etc. Get your head wrapped around the code.

2 Creating a page swapping mechanism

Notice that the new *backing store* device is not used. This code will compile and run, but the *real* memory is small.[2] Any program that uses 1 MB or more will fail.

Your task is to make use of the backing store device to swap pages to and from *real* memory. Each time you do, the page tables must be updated to reflect the change. Simulated pages backed

[1] Notice, also, that `mmu_translate()` now has a second parameter that indicates whether the memory reference is a *read* or *write* operation.

[2] Of the 5 MB in the default *real* memory size, the first 4 MB + 4 KB are reserved for the page table; slightly less than 1 MB is available for backing *simulated* pages.

by real memory should have their *resident* bit set and their translations should succeed; those not backed by real memory, and held only in the backing store, should have this bit cleared so that translations fail. The `vmsim_map_fault()` function must identify attempted uses of pages on the backing store and initiate a page swap. How you choose to approximate the *least recently used* policy in order to select a page in real memory for replacement is up to you.

3 How to submit your work

Submit your `vmsim.c` source code file using one of the two usual tools:

- **Web-based:** Visit the submission system web page.
- **Command-line based:** Use the `cssubmit` command at the shell prompt on `remus/romulus`.

This assignment is due on Friday, Nov-16, 11:59 pm.