Operating Systems Laboratory
Spring Semester 2017-18

Assignment 5: *Implement a Simulator for Virtual Memory Management*

Assignment given on: March 26, 201
Assignment deadline: April 02, 2018

Implement a simulator for virtual memory operations in a demand paging memory management scheme, which maps a virtual address space comprised of virtual pages onto physical page frames. The number of page frames in memory can vary, and will be an input parameter to your program. Only one virtual address space will be considered in this assignment (i.e. only one running process), and the size of the virtual address space will be 64 pages.

The input to the program will be a file containing a sequence of *instructions* and optional *comment lines* as shown below.

```
# Example page reference trace
# These are optional comment lines
1 25
0 45
0 18
1 17
```

a) A comment line in the input file starts with a '#' and should be completely ignored by the simulator and not included in the instruction count.

b) An instruction line is comprised of two integers, where the first number indicates whether the instruction is a read (value = 0) or a write (value = 1) operation, and the second number indicates which virtual page is accessed by that operation.

You have to implement the logical to physical address mapping using a single-level page table. In addition to the frame number, each page table entry would consist of the following bits:

i) VALID, indicating whether a page is present in memory or not. If the bit is 1, the page is present.
ii) MODIFIED, indicating whether a page has been modified after it has been loaded in memory. If the bit is 1, the page is modified or dirty.
iii) REFERENCED, indicating whether the page has been referenced after being brought into memory. If the bit is 1, it has been referenced.

Each page table entry should be implemented as a single 32-bit value (and not as a structure of multiple integer values), with the different fields manipulated using bit operations. Initially, all page frames are maintained in a free list, with frame-0 being the first frame in the list, and all page table entries are zero-ed (i.e. pages not present in memory).

During the simulated execution of each instruction, you need to simulate the behavior of the hardware. That is, you must check whether the page is present in memory, and if not, locate a free frame, and bring in the frame. Then the virtual address of the instruction needs to be mapped to the frame.

You also need to implement page replacement to handle the cases where there are no free page frames available. The following page replacement algorithms are to be implemented:

a) First-in First-out (FIFO)
b) Random
c) Least Recently Used (LRU)
d) Not Recently Used (NRU)
e) Second chance (derivative of FIFO)

When a virtual page is replaced, it must be unmapped and paged out. While you are not actually implementing these operations in the simulator, you still need to track them to generate the final statistics. Some sample output(s) to be generated is shown below:

Illustration 1:

Suppose that instruction on line 75 is a read operation on virtual page number 2. The replacement algorithm selects virtual page 42 or physical frame 26, and hence we first have to UNMAP the virtual page 42 to avoid further access, and write it back to swap device as the page was dirty (suppose). Then it pages in the virtual page number 2 into the physical frame 26, and sets the valid bit to 1 for the page table entry.

```
Instruction on line 75: 0 2

Output generated:
  75: UNMAP  42  26
  75: OUT    42  26
  75: IN     2   26
  75: MAP    2   26
```

However, if the page being replaced was not dirty, the output generated would be:

```
Output generated:
  75: UNMAP  42  26
  75: IN     2   26
  75: MAP    2   26
```

In addition, the simulator needs to compute and print the summary statistics related to the virtual memory management. This means it needs to keep track of the number of page faults, number of page transfer operations. It will also give the overall execution time in cycles, assuming that MAP and UNMAP operations each cost 250 cycles, page-in and page-out operations each cost 3000 cycles, and each memory access (read or write) costs 1 cycle.

The page table and information about which page is loaded into which frame should be printed at the end of the execution. In a trace mode, it can be printed after every instruction execution.

Note that for the NRU algorithm, the reference bits can be reset every 10^{th} page fault before carrying out the replacement operation.

Submission Guideline:

- Create the program as a single file as **`Ass5_<groupno>.c`** or **`.cpp`**, and upload it.