CSE 325 **Spring 2021**

Computer Project #10

Assignment Overview

This assignment focuses on memory management within an operating system. You will design and implement additional functionality to extend the previous project, as described below.

It is worth 50 points (5% of course grade) and must be completed no later than 11:59 PM on Thursday, 4/8.

Assignment Deliverables

The deliverables for this assignment are the following files:

`proj10.makefile` – the makefile which produces **`proj10`**
`proj10.student.cpp` – the source code file for your solution

Be sure to use the specified file names and submit them for grading via the CSE Handin system before the project deadline.

Assignment Specifications

The program will simulate the steps to manage primary storage using paging without a TLB. The system to be simulated contains 65,536 bytes of RAM which is divided into 64 page frames. Virtual addresses are 14 bits in length. The system uses LRU page replacement.

1. The program will simulate the actions performed in the memory management unit by processing a file which contains zero or more memory references. Each line of the file will contain the following information:

a) operation ("RD" for "read" or "WR" for "write")
b) virtual address (four hexadecimal digits, with leading zeroes)

Items in the line will be separated by exactly one space, and each line will terminate with a newline. For example:

```
RD 3bd8
WR 0ac2
```

The first line represents an operation where the process is reading from virtual address 3bd8, and the second line represents an operation where the process is writing into virtual address 0ac2.

2. For each memory reference in the file, your program will display one line with the following information:

a) operation being performed (two characters; "RD" for read and "WR" for write)
b) virtual address being referenced (four hexadecimal digits, with leading zeroes)
c) page number (one hexadecimal digit)
d) page offset (three hexadecimal digits, with leading zeroes)
e) page fault flag (one character; 'F' for page fault, blank otherwise)
f) write back flag (one character; 'W' for write back, blank otherwise)
g) physical address (four hexadecimal digits, with leading zeroes)

Items in the line will be separated by exactly one space, and the line will terminate with a newline. For example:

```
RD 00f6 0 0f6     94f6
WR 27a4 9 3a4 F W 8ba4
```

An appropriate error message will be displayed for each invalid memory reference.

Page fault: when V == 1 && P == 0

3. After the simulation is completed, your program will display the following counts (with appropriate labels):

a) total number of read operations
b) total number of write operations
c) total number of page faults
d) total number of write backs

4. After the simulation is completed, your program will display the contents of the page table. The display will contain one line for each page table entry:

a) index of the page table entry (one hexadecimal digit)
b) V bit (one character; '0' for not valid, '1' for valid)
c) W bit (one character; '0' for not writable, '1' for writable)
d) P bit (one character; '0' for not present, '1' for present)
e) R bit (one character; '0' for not referenced, '1' for referenced)
f) M bit (one character; '0' for not modified, '1' for modified)
g) frame number (two hexadecimal digits, with leading zeroes)

The page table display will include appropriate column headers. For example:

```
     V W P R M FN
----------------
[0]: 1 0 1 1 0 23
[1]: 1 0 0 0 0 00
 .
 .
[e]: 0 0 0 0 0 00
[f]: 1 1 1 1 1 24
```

5. Command line arguments will be used to specify the name of the file which contains the memory references to be processed (the "-refs" option), the name of the file which contains the process information (the "-proc" option), and to display debugging information (the "-debug" option).

a) The "-refs" option will be followed by the name of the file which contains zero or more references (as defined above).

b) The "-proc" option will be followed by the name of the file which contains one or more lines with information about the set of valid pages in the virtual address space of the process. The first line of the file will contain the number of page frames allocated to the process (in decimal). The page frames will begin with frame 20 (hexadecimal) and will be consecutive. For example, if 5 page frames are allocated to the process, the initial free frame list will contain:

0x20, 0x21, 0x22, 0x23, 0x24

where 0x20 is at the head of the free frame list and will be the first frame allocated.

All subsequent lines of the file will contain the following information:

a) page number (one hexadecimal digit)
b) write permission (one hexadecimal digit; 0 for not writable, 1 for writable)

Items in the line will be separated by exactly one space, and the line will terminate with a newline. For example:

```
4
2 0
8 0
9 1
```

6. If the "-debug" option has been selected, your program will display:

   a) the contents of the page table at the start of the simulation
   b) the contents of the page table after each memory reference is processed

7. The program will include appropriate error-handling.

Assignment Notes

1. As stated above, your source code file will be named "proj10.student.cpp" and you must use "g++" to translate your source code file in the CSE Linux environment.

2. Valid executions of the program might appear as follows:

```
proj10 –proc proc1 -refs test1 -debug
proj10 –debug –refs fileA –proc fileB
proj10 –proc test3p –refs test3r
```

3. Your program may assume that the "-refs" and the "-proc" files are formatted correctly:

a) If the "-refs" file can be accessed, the contents will be valid as per #1 under "Assignment Specifications".

b) If the "-proc" file can be accessed, the contents will be valid as per #5 under "Assignment Specifications".

4. Your program must create a data structure representing the page table and set all of the entries to zero at the start of the simulation.

Your program will then use the "-proc" file to initialize the page table entries for the pages which are part of the virtual address space of the process.

For this assignment, your program will actively manage the page table (and thus the page table contents will change over time).

5. Note that example output shown in #2 and #4 under "Assignment Specifications" is only intended to illustrate the format of the output.