CSC3002F: Operating Systems:

OS1: Assignment

Memory Management

Department of Computer Science

University of Cape Town
South Africa

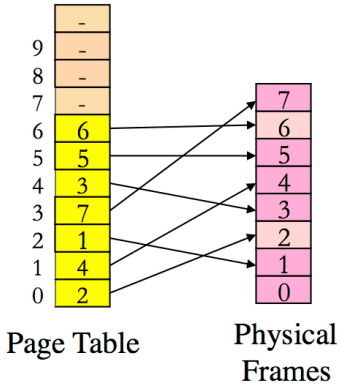

Figure 1: The page table of a process is as shown – Note only a few entries of the process page table are shown, as we will only use the first seven entries in this assignment.

Problem Description & Task

This assignment is about *OS* virtual memory management. Although, typically, an *OS* can support more than one process, this assignment assumes there is only one process. The assignment is to simulate memory management (specifically memory address mapping), where we assume a tiny computing system that supports up to $1KB$ physical memory with *12-bit* virtual addresses (4096 bytes). Suppose that the size of a *virtual page* and a *physical frame* is 128 bytes (i.e., with 7 address bits representing page offset).

Write a *Java* program called $OS1Assignment.java$. This program will take only one parameter – the input sequence filename, which contains a sequence of virtual memory accesses – where each address is stored as 8 bytes (*unsigned long type*). Your program should read and analyse each *virtual* address (in binary format) and translate it into the corresponding *physical* address (in binary format) based on the given page table, as shown in figure 1.

Note: to simplify things, you can place the fixed page-to-frame mapping (as shown in figure 1) into an array, before performing any address translation.

To verify that you can read the correct sequence of memory accesses, you can test your program with the given simple test sequence file ($OS1testsequence$). For example:

- First address should be: $0x00000044$, and the physical address (after translation) should be: $0x144$.
- Second address should be: 0x00000224, and the physical address (after translation) should be: $0x01A4$.
- And so on ...

For each address in the sequence file ($OS1sequence$)[1], use the given page table (figure 1) to perform the address translation, and generate a corresponding physical address. Since we are representing each physical address using *64-bits* (8 bytes), each physical address should use 8 bytes (as an *unsigned long*).

However, *note:* your output file (*output-OS1*) must contain each physical address converted to a *string* and thus just contain a *list of strings*.

After testing your program with the simple test sequence ($OS1testsequence$) file (which contains only 8 memory accesses), use the sequence file ($OS1sequence$) as the input file for the address sequence that will generate *translated physical addresses*, and place these physical addresses in an output file called *output-OS1*.

Submission Guidelines

Place, in a ZIP file named as your student number (e.g. PRTPIE003.ZIP), the following:

1. Your source code: $OS1Assignment.java$
2. Your output file: *output-OS1*
3. A README file that includes your name and student number and a one sentence description (<50 words) of the approach your algorithm uses for this memory address translation task.

Upload your ZIP file to *Assignments* tab on *Vula* by: **5.00 PM, 27 May, 2021**.

* * *

[1] *Note:* This sequence file: *OS1sequence* is different from the test sequence file: *OS1testsequence*. The latter is used to just test your code whereas the former is used to generate output for your assignment.