Designing a Virtual Memory Manager with Page Table

Overview:

This project consists of writing a program that translates logical to physical addresses. The RAM is a simulated by an array of 2^7 bytes (128). The page size (also frame size) is 2^4(16) bytes. The RAM contains (128/16 = 8 frames)

There is only one process and it has a logical address space of 2^9 = 512 bytes. The number of pages in the process is (512/16) = 32. Each address contains data that is an alphabet that fits in 8 bits (1 byte). The sequences of logical addresses generated by the CPU when running this process is given in a file called "addresses.txt".

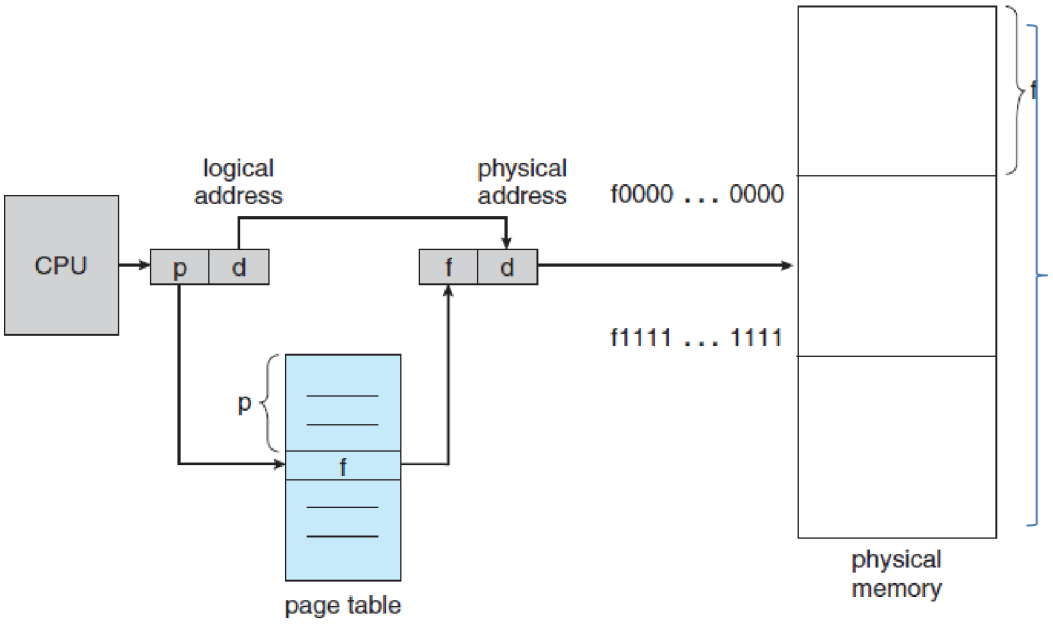
The entire process is available on the disk which is simulated by a file called PROCESS.txt. Note that PROCESS.txt acts like swap space of the disk. Actual data at each address is one byte of type char and data can be found in PROCESS.txt.

Simulate the page table as an array with 32 entries. Initially, all entries in the page table will be invalid since there are no pages on the RAM. When CPU needs data at logical address 500 (first time) there is a page fault and the page containing the logical address 500 is bought into the simulated RAM and placed in a frame. The logical address is then translated to physical address and the data is read from the RAM. The second access to addresses in the same page will not cause page fault unless that page has been removed.

The sequences of 300 addresses that CPU needs in available in addresses.txt. Your program will read the logical address from addresses.txt and find the page number and bring in the page from PROCESS.txt and put it on RAM and then read the data at the address. Of course if the page is already available on the RAM, the frame number will be available in the page table.

When the RAM is full and no more pages can be brought in, least recently used algorithm is used remove a page from the RAM.

The address translation scheme is shown in the figure below.



Your program is to translate each logical address in the input file to a physical address and determine the contents of the signed byte stored at the correct physical address. Your program is to output the following values:

1. The logical address being translated (the integer being read from addresses.txt).
2. The corresponding physical address (what your program translates the logical address to).
3. The data which is a byte value (letter) stored at the translated physical address.

For example,

Address	Physical Address	Data
500	312	b

After completion, your program is to report the following statistics:
You will list the list of pages that faulted
Page-fault rate—percentage of address references that resulted in page faults.

Internal Requirements

You cannot read data directly from PROCESS.txt into your program. You must bring in the entire page and then read data from the RAM

This program is taken from the textbook by Silbershatz, Galvin and Gagne-Operating Systems