IT308: Virtual Memory – Exercises and Prog. Assignment

For this lab you will be mostly using the virtual memory simulation features of Camera, a cache and virtual memory simulator. You may also find the cache simulations interesting, however we won't be working with those here. To get checked off on this lab, you will have to solve Exercises 1 to 3 below, and show your work to a TA.

Once Camera opens up, select the virtual memory option to open a visualization of the virtual memory system. In the top left you can see the contents of physical memory. Just below that is a listing of all the pages of virtual memory for this process. To the right of these items are the contents of the TLB and the Page Table. At this point these should all be empty as we haven't done anything yet. Read about the statistics of your memory system in the "PROGRESS UPDATE" box at the bottom of the window. This area will keep you updated on your status through the simulation as it progresses. You can move the simulation forward, backward or start it over from the beginning using the buttons to the right of the "PROGRESS UPDATE" box.

This handout also includes a description of the second programming assignment in which you will implement several paging policies. This assignment is to be submitted via moodle and is due on 10th February, 23:50 hrs.

Exercise 1: A Sample Run

Click the button labeled "Auto Generate Add. Ref. Str." at the right-hand side of the window. This will generate a set of ten address references. You can think of these as a series of x86 "load word" instructions reading from the memory address specified. Click the button labeled "Next" to begin the simulation.

For the rest of this exercise you are at the mercy of the "PROGRESS UPDATE" box. After each click of the "Next" button examine the contents of the box and the current state of the memory system. Try to really get an understanding of what is going on in the TLB, the Page Table, and Physical Memory at each step.

Once you have reached the end of the simulation note the number of TLB Hits and Misses and Page Hits and Faults. Write these numbers down, along with the sequence of memory accesses used to show to your TA during checkoff.

Show your TA the number of TLB Hits and Misses and Page Hits and Faults, along with the sequence of memory accesses used. Did you have any Page Hits? Why or why not? Be ready to describe the steps on a memory request.

Exercise 2: Miss

Now that you've seen what a random workload looks like in the VM system let's try creating a custom workload with a specific property. Your goal for this exercise is to create a workload of ten memory accesses that will cause ten TLB misses and ten Page Faults. You should be able to come up with such a workload on paper, but then you should run it in CAMERA to verify your work. You can specify a custom workload in CAMERA by clicking the button labeled "Self Generate Add. Ref. Str." and entering in the addresses you want to reference one at a time. When you are satisfied that you've got a valid sequence write it down and be ready to show it to your TA during checkoff.

Show your TA a workload of ten memory accesses that will cause ten TLB misses and ten Page Faults.

Exercise 3

Given your sequence of memory accesses from Exercise 2, can you find a change to a single parameter (e.g. TLB size, page table size, memory size, etc.) that would result in the same number (ten) of TLB misses but result in fewer than ten page faults? Work through this on paper and be ready to show your results to your TA during checkoff.

Explain the single parameter change that would result in ten TLB misses, but fewer than ten page faults.

Programming Assignment 2: Page replacement simulator

In this assignment, you will write a program that simulates the virtual memory page replacement algorithms: Optimal (OPT), Least Recently Used (LRU) and First-In-First-Out (FIFO).

This assignment contains three separate programs: a page replacement simulator vmsim, a page reference generator vmgen, and a page statistics program vmstats that produces a consolidated data file used for performance plots.

Section 1: Specifications of the page replacement simulator vmsim

- vmsim must accept three command-line arguments in the following order: (a) the total number of physical memory frames (maximum 100), (b) an input filename where a sequence of page references is stored, (c) the chosen algorithm. The three possible values for (c) are the strings opt, lru and fifo. (If vmsim is run with the wrong arguments or no arguments, it should print out usage instructions and exit.)

Example:
>vmsim 5 vmrefs.dat lru

- The format of the input file must be a simple ASCII sequence of integers in the range 0 to 99 separated by spaces, for example: 51 7 34 0 8 45 21. No symbols or formatted text, just a pure sequence of space-separated integer numbers. This file can be created by hand or generated by the vmgen program (see section 2 below).

- vmsim will first read all the memory references from the input file and store them in a local array. Then, it will play back these references one by one and print out for each reference the current allocation state of physical memory frames in the following format:

  34: [51| 7|34| |]

- This line means that after using page 34, frames 0, 1 and 2 are occupied by pages 51, 7 and 34, and frames 3 and 4 are empty. Frames must start with open square bracket [, end with closed square bracket] and be separated with vertical bar |. One-digit page numbers should have an extra space to the left so that frames are always 2 characters wide (2 spaces for an empty frame). Each page fault should be signaled by an F character two spaces to the right of the closed bracket, for example:

  45: [45| 7|34| 0| 8] F

- After processing all the memory references, vmsim should finally print the total number of page faults and the miss rate (page faults divided by number of references). It should start counting page faults and page references only after all frames have been initially filled, e.g., with 3 frames that means starting at the 4th step. Use this printout format:

  Miss rate = 237 / 997 = 23.78%

Section 2: Specifications of the page reference generator vmgen

- vmgen must accept three command-line arguments in the following order: (a) the range of page references (maximum 100), (b) the length of the sequence and (c), the name of the file that will be generated. (If vmgen is run with the wrong arguments or no arguments, it should print out usage instructions and exit.) Example:

  >vmgen 10 200 vmrefs.dat

- vmgen will then generate a sequence of the desired length containing random page numbers uniformly drawn between 0 and the range minus one (i.e., 200 page numbers between 0 and 9 in the example above). Important: no page number in the sequence should be equal to the number that precedes it (hence, follows it, too). For example: 2 7 7 0 0 0 3 3 … is not a valid sequence but 2 7 0 3 … is. vmgen must write this sequence into the file given in input.

Section 3: Specifications of the page statistics program vmstats

- vmstats must accept four command-line arguments in the following order: (a) the minimum number of frames (no less than 2), (b) the maximum number of frames (no more than 100), (c) the frame number increment (positive), (d) the input filename containing the references. (If vmstats is run with the wrong arguments or no arguments, it should print out usage instructions and exit.) Example:

  ```
  >vmstats 5 40 10 vmrefs.dat
  ```

- vmstats will loop over the three (grads: four) page replacement methods: opt, lru and fifo and for each method, it will loop over the number of frames, starting at the mininum and applying the increment until it exceeds the maximum (i.e., 5, 15, 25, 35, in the example above). For each method/number of frames pair it will calculate the page fault rate using the reference file given in input and print out a one-line message containing this rate. Use this printout format:

  ```
  lru, 15 frames: Miss rate = 237 / 985 = 23.78%
  lru, 25 frames: Miss rate = 163 / 975 = 16.35%
  ...
  ```

- Therefore, the implementation of both vmsim and vmstats must rely on a common utility simulation engine (entry-point function) with the difference that vmsim is going to run the simulation once and in verbose mode (displaying all the allocation steps and the final miss rate; see 1.), whereas vmstats is going to run the simulation repeatedly and silently (displaying only the final miss rate for each pair in the loops). In accordance with this design, please keep the simulation engine in a source file distinct from the vmsim and vmstats capstone code.

- As a by-product, vmstats must also generate one consolidated results file containing the matrix of all the calculated miss rates. The results file must be named vmrates.dat and must be formatted in the following way: it should contain exactly four lines of N space-separated numbers, where N is the number of numbers of frames (4 in the example above). The first line must be the sequence of numbers of frames (5 15 25 35 in the 3 of 3 example above), the other three lines must be the sequences of rate values for each number of frame. These sequences must be in the following order: opt, lru and fifo. vmrates.dat should not

contain formatted messages or symbols, just four lines of pure space-separated numerical sequences with line feeds between sequences.

Submission and Required Files

Follow moodle page.