Lab 8: OS, Virtual Memory

Deadline: Friday, November 5, 04:00:00 PM PT

Goals

- Explore the workings of virtual memory, specifically the TLB and the Page Table.
- Analyze TLB hit rate and Page Table hit rate and determine what accesses optimize these values.

Setup

You do not need to pull from the lab starter for Lab 8.

Info: Virtual Memory

Virtual **M**emory is an important concept in computer architecture and operating systems. It provides the needed abstraction for more than a single process to be run on a system at once. Though at this point, we don't yet have direct support to simulate what accesses might look like **without** virtual memory, we can visualize it through memory address breakdown.

The way in which virtual memory addresses are broken down looks arbitrary, but they follow a precise structure. Each component of the address breakdown addresses how large the corresponding target is; as we increment the addresses, we cross each until we reach the next block at which point it is indexed separately. It functions very similarly to how caches work in the cache TIO breakdown; in virtual addresses, its broken down as different levels of index bits before we reach the **O**ffset bits.

Exercise 0

You will be submitting your answers to the Lab 8 Google Form

For this lab we will mostly be 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. Unfortunately, Camera is known to have issues when trying to run it on the hive machines or Linux machines, so it's recommend to download Camera from this link, and simply double click on the jar file to run it on your own (non-Linux) laptop. If you are using a Linux computer or double-clicking does not work, to run the jar file in the command line navigate to the directory you downloaded the jar file and run

```
$ java -jar Camera.jar
```

If you're on a Mac, you may need to go to "Security & Privacy" in your settings and click "Open Anyway" to allow Camera to run. Some displays don't seem to play nice with the standard Camera app. If the values in memory are all squished together, try running this version of Camera. If you are unable to find a way to get Camera working on a machine, please partner up with someone who does.

Once Camera opens up, select the "Virtual Memory and Paging" 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.

Before you continue, **MAKE SURE THAT YOU OPENED THE VM SIMULATOR AND NOT THE CACHE SIMULATOR**.

Exercise 1: Working with CAMERA

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 RISC-V "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.

Checkpoint

Please answer the following questions in the lab form.

1. Given the way the address was broken down, how big (in words) are the pages in this model? Leave out the units in your answer (e.g. 4 instead of 4 words).
2. Did you have any Page Hits? (Why?) Can you think of a set of ten addresses that would result in a Page Hit? Your answer should be two [yes/no]'s separated by a comma.
3. What is the process by which we access memory given a virtual address on the very first access assuming a page fault? Order the steps given below. Give your answer as numbers delimited by by a comma with a space, i.e. [num], [num], ... for example, `1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12`.
    1. We update the page table to map the corresponding VPN to the PPN.
    2. Calculate number of virtual page number (VPN) bits through the VA bits and offset bits.
    3. TLB does not contain the VPN so we access the page table for the corresponding VPN.
    4. We access the corresponding word using the offset.
    5. We bring the corresponding virtual page into physical memory from disk.
    6. The page table's entry for VPN has a valid bit of 0.
    7. Get VPN by taking `VA[address bits - 1 : offset bits]`.
    8. Calculate number of offset bits by taking log 2 of page size.
    9. We update the TLB with the corresponding PT entry.
    10. Get offset by taking `VA[offset bits - 1 ; 0]`.
    11. Access TLB for corresponding VPN.
    12. Access given virtual address.
4. How many PPN bits are there?
5. How many VPN bits are there?
6. How many physical pages are there?
7. How many virtual pages are there?

Exercise 2: Misses

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. We will not be autograding this exercise, but you should still work this problem out for practice.

▶ Hint

▶ Example Solution

Exercise 3: Fixing our Faults

Given your sequence of memory accesses from Exercise 2, can you find a change to a single parameter (e.g. TLB Size, Physical Memory Size, Virtual Memory Size, Page Size) that would result in the same number (ten) of TLB misses but result in fewer than ten page faults?

Checkpoint

1. Write down two parameters for which if each got individually changed (while all other parameters stay the same) would result in ten TLB misses but **fewer** than ten page faults. Format your two answers in alphabetical order as such: `A_ANS, B_ANS`. For example, if your answers are `daisy, cotton ball`, write it as `cotton ball, daisy`.

Exercise 4: Bringing it All Together

Watch this video of a VMSIM simulation:

Observe what is happening and answer the following questions:

What is different about the setup of the system in this question as compared to the setup in CAMERA? In particular, what are P1, P2, P3, and P4? If you watch closely you'll see that this simulation reports a much higher percentage of TLB misses than random runs on CAMERA did. Why might this be? (If you have trouble following the simulation, use the `appletviewer` and turn down the speed using the slider on the bottom right.)

Checkpoint

1. Select all factors that would cause the simulation to show a lower TLB hit rate than our original model.
    1. A single process is running.
    2. Multiple processes are running.
    3. Running multiple processes requires the system to context switch.
    4. Running a single process requires context switches.
    5. Context switches preserve TLB state.
    6. Context switches invalidate the TLB.

Exercise 5: Feedback Survey

We are working to improve the labs for next semester, so please fill out this survey to tell us about your experience with Lab 8. The survey will be collecting your email to verify that you have submitted it, but your responses will be anonymized before the data is analyzed. Thank you!

Submission

Save, commit, and push your work, then submit to the **Lab 8** assignment on Gradescope. The autograder will check to see if you have submitted the feedback form and received full credit on the answer form.

Checkoff
