Virtual Memory and Caches

ECE598: Advanced Operating Systems – Homework 7

Spring 2016

Due: Thursday, 7 April 2016, 9:30am

This homework involves virtual memory and caches.

1. **Download the homework code template**

   - Download the code from:
     `http://web.eece.maine.edu/~vweaver/classes/ece598_2016s/ece598_hw7_code.tar.gz`
   - Uncompress the code. On Linux or Mac you can just
     `tar -xzvf ece598_hw7_code.tar.gz`

2. **Modify the code to gradually improve memset() performance (8pts)**

   We will be testing the speed of the `memset()` routine which is found in `string.c`. You'll notice that it's not very optimized, just a byte-by-byte copy.

   (a) I've provided the benchmark routine, `run_memory_benchmark()` that runs the memset over 1MB of memory 16 times. It measures before and after with the 64Hz timer tick and prints how many ticks have elasped.

   Report the performance in the README: the number of ticks, and also calculate and report the number of MB/s.

   (b) Now, enable the L1 instruction cache.
   In `kernel_main.c` find the "Running memory benchmarks" printk and put your code there.
   Add a call to the `enable_l1_icache()` routine (that's in the provided mmu.c code)
   Add a call to `run_memory_benchmark()`.

   Report in the README how many ticks elapsed, as well as what this is equivelent to in MB/s.

   (c) Next enable the branch predictor (in addition to the L1 icache, i.e. leave the previous test in place). Use the `enable_branch_predictor()` routine and report the ticks and MB/s.

   (d) Next also enable the L1-dcache. This is tricky, as you will need to enable virtual memory as well as the dcache. Luckily for you I've provided code for this. Run `enable_mmu(0,memory_total)` as well as the `enable_l1_dcache()` routine and report the ticks and MB/s.

3. **Something cool (1pts)**

   (a) Our memset() routine is very inefficient. Can you write a better routine that runs faster?
   Report what method you chose and its performance in ticks and MB/s.
   The easiest way is to just copy data in integer (32-bit) sized chunks. Harder ways include using the STM assembly language instructions, or possibly using complex NEON instructions.

4. **Questions (1pt)**

   Answer these questions in the README file.

   (a) Name one feature found in Linux/UNIX that is made possible (or easier) because of virtual memory.

   (b) Why do operating systems have filesystems? Could you design a system that did not use filesystems at all?

5. **Submit your work**

   - Run `make submit` in your code directory and it should make a file called `hw7_submit.tar.gz`. E-mail that file to me as well as the document with the answers to the questions.