ECE 5720, Fall 2021
Virtual Address Translation
Assigned: Nov. 16, Due: Nov. 30

1 Introduction

In this lab, you will write a c++ program to translate a virtual address into a physical address. Virtual addresses are an important abstraction in modern computer systems. Without virtual addresses, every program has to be aware of other program's memory usage and not interfere with each other.

2 Address translation

You will essentially implement the first part of Practice problem 9.4 from the book pg 824. It is highly recommended that you complete that example and read through the previous section where virtual memory is explained.

The virtual address translation can be devided into three parts. This is the order I would recommend trying to complete this assignment.

Physical Page to Virtual Page

The first step in translating a virtual address is converting the virtual page number to a physical page number. You will have to translate the PAGE_SIZE constant into a number of bits of page offset. The remaining bits are the virtual page number. Use the sizes defined in constants.h and the page table in the array pageTable. The page table array contains a list of physical page numbers for each page table address.

TLB Lookup

The next step in the assignment is to check if the virtual page number is in the TLB BEFORE cheching the page table. The TLB is technically a cache for the page table so address translation can occur completely without the TLB. The TLB is used to speed up page table access by caching recently accessed entries in the page table. You don't need to implement a cache again for this assignment, just lookup page numbers in the TLB and use the TLB entry if it is found. Your program should print "TLB HIT" or "TLB MISS"

depending on if the page number was found in the TLB. The tlb is stored in an array of tlbEntries called simply tlb. The sizes and associativity of the tlb are defined in the constants.h file.

Page Fault

The last thing your program should do is print "PAGE FAULT" and exit should a page table entry be invalid. A page table entry is valid if the corresponding element of pageTableValid is true.

3 Handout Instructions

> Please download the virtual-memory.tar file from the course wiki page. It's located in the Homework Schedule section.

Start by copying `virtual-memory.tar` to a (protected) directory on a Linux machine in which you plan to do your work. Then give the command

```
unix> tar xvf virtual-memory.tar
```

This will cause a number of files to be unpacked in the directory. The only file that you will be modifying and turning in is `virtual.cpp`

You will evaluate the performance of your address translations using the two example constants included in the example1 and example2 folders

To run your address translator with one of the example constants, simply copy the contents of the folder out and make

```
unix> cp example1/* .
unix> make
unix> ./virtual 03d4
...
```

4 Logistics

This project can be done individually or in a group of 2 students. All handins are electronic. You should work only on your SSD device.

5 Evaluation

Your score will be computed out of a maximum of 100 points based on the following distribution:

30 Correct translation of virtual address to physical address

30 Correct determination of TLB hit or TLB miss

30 Correct detection of Page faults

10 Coding style and commenting.

Autograding your work

The autograder script will test your program on examples similar to the examples used to grade this assignment.

```
unix> ./autograde.sh
```

6 Handin Instructions

Please submit `virtual.cpp` through Canvas. Make sure your code compiles using the Makefile provided in the handout tar package. The TA will evaluate the performance PLEASE DO MENTION the name of your teammate in the comment section during the submission. Every student needs to make a submission.

7 Hints

- Program output should contain the physical address in hex, either ”TLB MISS” or ”TLB HIT” , and ”PAGE FAULT” if there was a page fault. You are welcome to print other debug information but that is all the autograder looks for
- Use the book Practice Problem 9.4 (Only A-C) and the corresponding parts of chapter problems starting at 9.11. Don’t worry about the cache index, tag, hit, etc for this assignment.
- Funcitons to print the TLB and Page table are included in the handout purely for your convienence in debugging if you would like to use them.
- Don’t modify the make file because the grading will be done with an unmodified makefile.