CS2610
Lab 8 : Handling Instruction and Data Page Faults

1 Introduction

Now that you are familiar with page tables from Lab 7, the goal of this lab is to handle paging on demand. A Page fault occurs when a process tries to access memory that is not mapped to a physical page in main memory. The OS brings the corresponding page into main memory and creates a corresponding page table entry. In this lab, on a page fault, you will:

1. Swap-in the page, i.e., transfer data to the physical page.
2. Update the page table.

2 Problem

In the given code template, the page tables are initialized in supervisor mode as seen in lab 7. The user code will generate page faults. Your task is to handle ALL of the page faults generated by the user code in machine mode by writing assembly code in the page fault handler section only. Make sure you consider the appropriate CSRs like mcause, mtvec, and mtval.

1. The user code and user data lie in the pages beginning from the physical address 0x80001000 and 0x80002000 respectively.
2. The page table is initialized with the mapping of virtual address 0x00000000 to the user level code section and 0x00001000 to the data section.
3. Instruction page fault: Allot an available physical page and swap in the code. Assume every instruction page contains the same code as user code. Hence you will have to copy the contents of the page beginning from 0x80001000 into the new physical page. (refer Figure 1)
4. Data page fault: Map the fault-generating virtual page to the physical page of the user data page starting from address 0x80002000. (refer Figure 2)
5. Handle the Level 0 page table entry indices dynamically in both the cases.

Note: DO NOT hardcode the values. Your code should be able to handle any page-faults generated by the user code in the virtual space 0x00000000 to 0x00010000.

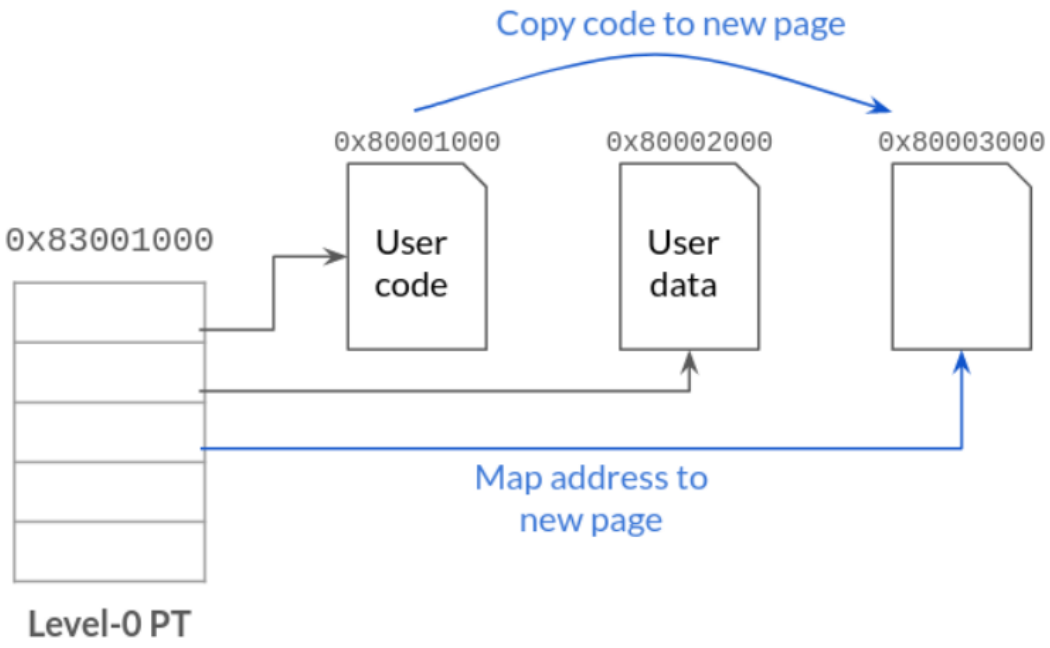

Figure 1: Instruction Page Fault

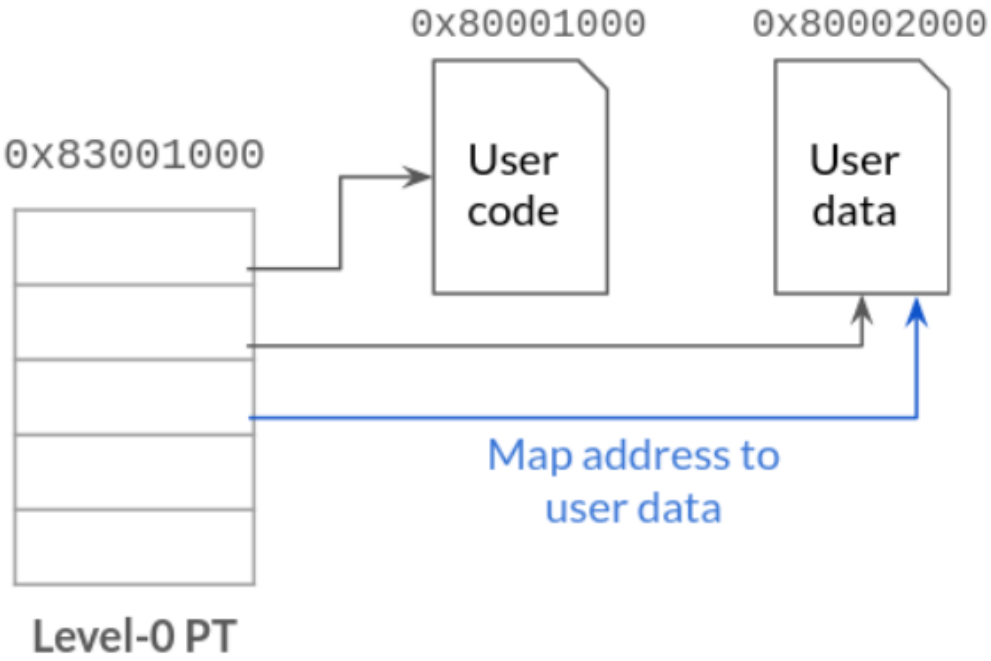

Figure 2: Store Page Fault

3 Submission

The code template is in the file template lab8.s in the zipped folder. The code alterations must be done only in the page fault handler section of the template file. You are expected to submit the following:

1. Your assembly code file.
2. The corresponding objdump file.
3. Screenshots of spike simulation showing, for each instruction page fault:
   (a) address of var count stored in register t1.
   (b) value of var count stored in register t2.
   (c) jump address stored in register t3.

Compile your code using:

```
$riscv64-unknown-elf-gcc -nostartfiles -T linker.ld <your_program>.s
```

Note:

1. All files should be submitted in a zipped folder.
2. The zipped folder should be named $< Roll_No >$_Lab8.zip.

Resources

https://people.eecs.berkeley.edu/~krste/papers/riscv-privileged-v1.9.pdf