Operating Systems
CSCI-UA.0202 Fall 2013

Programming Assignment 3
The Page Table

Due Monday, December 9

Note: This is not a group project. Each student must write his or her own code and not use the code from other students.

Also: Do not start this assignment until you have finished and submitted your tlb.c code from the second programming assignment.

This assignment is to implement a two-level page table within the simulated memory management system that you used for Project 2. You do not need to download my code again.

The details concerning the page table are described in the files `page.h` and `page.c`. You are welcome to use the code that is already in `page.c`, but you will need to fill in the rest of the needed code. `page.c` is the only file you need to modify. When you are finished, email me your `page.c` file.

The page table should be implemented as follows:

- The first level page table contains 1024 entries, each entry being a pointer that can point to a second level page table. The procedure `pt_initialize_page_table()`, which you should implement, should create the first level page table using `malloc()` and set every entry to `NULL`.

- A second level page table also contains 1024 entries. Each entry of a second level page table should contain the following elements:

    - present bit (1 bit)

    - page frame number (20 bits)

  The entry itself can be represented by a single 32-bit word (i.e. an unsigned int) and masks can be used to extract the present bit and the page number.

- Second level page tables should only be created when necessary. The procedure `pt_update_pagetable()`, which inserts a page table entry for a specified page, should create the appropriate second level page table (using `malloc()`) to hold the entry if it does not already exist. The appropriate entry of the first level page table should then be updated to point to the newly created second level page table.

- A page fault should occur when the procedure `pt_get_pageframe()` is called with a specified page number but either there is no second level page table to hold the entry for the specified page or if the present bit in the entry is 0. `pt_get_pageframe()`, in the case of a page fault, should signal that a page fault occurred by setting the variable `page_fault` to `TRUE`. Otherwise, it should set `page_fault` to FALSE and return the page frame number corresponding to the desired page.

- The MMU and kernel code, which I provide, will handle issuing the page fault interrupt, bringing the desired page into memory (and evicting a page, if necessary), and updating the page table. Your code does not have to worry about any of that.

You should compile your `tlb.c` from the previous project to `tlb.o` by doing:

```
gcc –m32 -c tlb.c
```

You only need to do this once.

Then, when you are ready to compile the entire program using your `page.c`, type

```
gcc –m32 –o memory page.c tlb.o cpu.o mmu.o kernel.o
```

Alternatively, use the Makefile provided by typing

```
make proj3
```

The executable file generated by the compiler is `proj3.exe` (using cygwin on a PC) or simply `proj3` (on a Mac or Linux machine). You should again compare the output of your program to that of `ben.exe` or `ben` .

Let me know if you have any questions.