CSCI 447 - Operating Systems, Spring 2024
Homework 1 : Chapter 1, First kpl program
Due Date: Check Canvas

~~Collaboration~~ Policy

Your homework submissions, for this and all future homework assignments, must be yours. You may chat with other students on a high-level about topics and concepts, but you cannot share, disseminate, co-author, or even view, other students' code. Please ask if this is unclear.
The use of the textbook as help is allowed. But, do not use prose from the textbook verbatim. Paraphrase. Rewrite in your own words..

Overview and Goal

This is the first of several homework assignments, and is an extension of Lab 1. **It is highly recommended that you complete lab 1 prior to starting the coding portion of this homework**.

1 Written/TF/Multiple Choice/etc. Questions : 24 points

Please answer the below questions for homework 1, in Canvas. Upload your answers as a single PDF. For all free-response questions, be concise, but not so brief nor cryptic that your answer is one or only a few words.

1. Select the **three** choices that best specify the main functional roles of an OS.
    - allocate resources
    - control and supervise the execution of multiple processes
    - provide an environment for executing programs
    - provide an interface between a user and hardware
    - compile and link programs
    - act as an intermediate between the user and hardware
2. It was mentioned during lecture that a main goal of the OS is to ensure that the CPU is always in use. In other words, the OS aims to make sure that compute resources are being utilized efficiently and maximally. Provide a concrete example when it is appropriate for the OS to forsake this aim, and to waste resources and/or permit the CPU to be idle even when some processes actively are in need of compute resources.
3. Which of the following instructions should be run in kernel mode? Select all such choices.
    - set value of a timer
    - read the clock
    - switch from kernel to user mode
    - clear memory
    - modify the entries in a device-status table
    - turn off interrupts
    - switch from user to kernel mode

4. Assume that cache is as inexpensive as a hard drive, or other secondary storage, such as a tape deck. Further assume that somebody has opted to build a computer that has just cache, but no hard drive nor other secondary storage. Explain why such a computer is impractical.
5. Explain how the two modes of a computer – kernel and user – function as basic security measures.
6. What is the purpose of interrupts, and how does an interrupt differ from a trap?
7. DMA enables accessing quickly the data on an I/O device, without much (sometimes none) CPU intervention. Assume that DMA is in progress, and that the CPU is executing some process X. Does this process X interfere with the execution of the user program that is using the DMA? Why or why not? If yes, what type(s) of interference is experienced?
8. Describe at least 3 distinctly different challenges that engineers face when developing an operating system specifically for a mobile device.

2 Coding Task : 20 points

This single coding task is an extension of Lab 1. This assignment serves as a refresher on C, and assembly code, which you must write.

1. Create a new branch on your gitlab for this class. Name the branch `homework1`. **When creating a new branch, unless you have a very good reason to do otherwise, you should do that from the master branch.**
2. `checkout` the new branch, and create a new directory, *homework1*. All work for this homework assignment should be done in the *homework1* branch and *homework1* directory.
3. Download the *homework1Files.tgz* file from the Files section of the Canvas course, and save it to the homework1 directory of your git repo. You'll need to transfer the files from your local machine, to your CS computer account. Use one of the many available free SFTP clients available online. Unzip and untar the files. Yes – the files are the SAME as those for Lab 1.
4. Add a new function, `GetCh`, to *Runtime.s*. The function should accept no arguments, and read a character from input, and return the character received (either as a char, or int, or ...). See *Echo.s* for examples of code for writing functions.
5. Add the external definition for `GetCh` to *System.h*. Make sure you export the new function in *Runtime.s*.
6. Create a new program, *aFunProgram.k*, which implements the same "echo" behavior as the *Echo.s* program, although it uses the function `GetCh`. When your program is running, and a user inputs *q* as the first (or only) character and presses enter, the program should stop and the debugger invoked. **If the first letter typed by the user is anything other *q*, the program should NOT jump into debugger mode. If the user types anything other than *q* as the first character, and presses return, the program should echo the character(s) input by the user, and continue running in non-debugger mode and wait for the next prompt**. **Hint: Read the code for `getCatchStack()` in *Runtime.s* to figure out how to return data as a value of a function.** *aFunProgram.k* is short for *aFunctionProgram.k*, but that should be obvious because *fun* and *function* are synonyms.

7. Alter the *makefile* file accordingly so that when `make` is invoked from the command line the aFunProgram executable is created.

8. Create a transcript of a terminal session in which you demonstrate the use of your program aFunProgram. Name the file *homework1-script* or *homework1-script.txt*.

9. `add`, `commit`, and `push` your new file, and changes to existing files, along with the script, to your *homework1* branch of your gitlab.

3 Rubric

Answers to written questions	24
Programming Task : setup: git branch & directory created correctly, no .o files, nor aFunProgram, pushed to git	2
Programming Task : `GetCh` written, with a return value. Commented, and added to *Runtime.s*, and its declaration added to *System.h*.	5
Programming Task : *aFunProgram.k* written, with comments, which implements use of `GetCh` to mimic the same behavior as *Echo.s*. `GetCH` should return a value, that is then used.	5
Programming Task : script file, either *homework1-script* or *homework1-script.txt*, created and uploaded	3
Programming Task : *makefile* correctly updated	2
Programming Task : *aFunProgram* runs correctly, and program jumps into debug mode ONLY if q is the first character types	3
Total	44 points