Lab Assignment 1
ZedBoard Linux Continued

The goal of this lab is to continue with your introduction to the Zedboard, write a more sophisticated C program on the Zedboard, and learn how the compiler manages different data formats.

Instructions

- Each lab assignment consists of a set of pre-lab questions, the actual time in the lab, and a lab report.
- The pre-lab assignments prepare you for the challenges you will be facing in the actual lab. So that each lab member gets the same benefits, pre-lab assignments have to be solved individually. Feel free to discuss problems with class and lab mates, but complete the assignment on your own.

Reading List

The following reading list will help you to complete the pre-lab assignment. The readings will also help you in subsequent lab assignments. Please complete the readings that are marked ***[Required]*** before attempting the pre-lab assignments.

[1] Try to remember and follow the programming standard rules listed below for all your programming tasks in this lab and future.

1. Every function should have a preamble that describes in a few sentences the purpose of the function, describes the input variables (type and range), the output variables (type and range), and describes any side effects (e.g., modification of global variables, memory allocation, etc.).
2. Every line of code should be commented.
3. Variable names should be descriptive and should be nouns. Variables names should be lower-case.
4. Function names should be descriptive and should be verbs. Function names should be lower-case.
5. Use proper indentation to clearly define control flow in your program.
6. Utilize structs whenever possible to build reusable data structures.
7. Functions should be limited in length to a single page of code.

Pre-Lab Assignment

We have only very limited time for the lab available. To make efficient use of the lab time, you will need to prepare for it. First, go through the assigned reading above and get an overview of the schematics and the manuals mentioned. Second, approach the questions below to apply your knowledge. Please submit the solution to the pre-lab assignments in PDF format via blackboard.

C Programming Preparation

Pre 1.1) Pass by value / Pass by reference

Two general means exist for argument passing into a function: pass by value and pass by reference. They differ in whether changes to the argument variables inside the callee (the function called) are visible in the caller (the function calling the callee function).

a. Assume the following function prototype:
```
void foo(int myVar);
```
Assume we call function *foo* from *main*. If *myVar* is changed within the function *foo*, is that change visible once returning to the function *main* that called *foo*?

b. Write a function prototype for a function "*bar*" that accepts as an argument a pointer to an integer *myVar*.

c. Within *bar*, write a C-statement that assigns the variable *myVar* a value "42". Is the effect of this assignment visible in the function calling *bar*?

d. Summarize the following question in a few sentences: What is the difference of "pass by value" versus "pass by reference?"

Pre 1.2) Next we consider using *printf* to print values in different formats.

Assume the variable *myVar* is declared as an *int*. Write a C-statement to print *myVar* as:

a. a decimal value
b. a hexadecimal value (base 16)
c. a octal value (base 8)

Pre 1.3) Write a program "*IntegerMath.c*" containing 4 functions for the basic operations "add", "subtract", "multiply", "divide". You are basically developing a simple calculator. Each operation should be implemented as a single function. Each function accepts 2 integers as input, and returns one integer output. Develop a *main* function that first prompts the user to input 2 positive integers (operands), and then asks the user to specify a math operation (+, -, *, /). Each of the 4 functions should be called from the *main* function to perform the appropriate math function and then returns the result. The *main* function should print out each result in decimal format. Consider any warning that should be issued to the user so that the calculator does not crash.

Submit the pre-lab solutions under the pre-lab link in Blackboard Assignments.

Lab Assignments

This lab continues to familiarize you with the Zedboard as an embedded computing device and C programming. In this lab you will get more accustomed to working in Linux, write more complex C programs and learn how to use some of the options available with the gcc compiler. Moreover, you will learn to use different data types, and use functions to modularize your code and make it easier to debug.

Connecting to ZedBoard

1. Login into the Windows desktop PC (we will refer to this system as the host) using your myneu credentials.
2. Make sure your Zedboard is powered (plugged in and the power switch on the board is in the ON position), and has been given time to boot up (generally takes a few minutes). Then connect the Zedboard to the PC host via Ethernet to the RJ45 connector.
3. Make an SSH connection to the Zedboard, where the IP address is 192.168.1.10, and port number of 22.
   - *Don't work as root, you already created your user accounts, use that!*

Lab 1.0 Compile IntegerMath on ZedBoard

5. Test the IntegerMath program from the pre-lab assignment on the ZedBoard. (You can use an SFTP connection to securely transfer files from the host to the ZedBoard)
6. Compile and run your IntegerMath program. To compile, use the gcc compiler. To run the program, you will need to provide the path of the program (e.g., ./IntegerMath) to allow the runtime system to find your program.
7. Record the file permissions associated with the source and executable files. Also record the size of each file.

Lab 1.1 Explore Compiler Option –S

8. Compile IntegerMath.c using the –S switch. Inspect the file that was created? Discuss what this file is. You can use the manual pages for gcc to see what the –S switch generates.
9. Now compile the IntegerMath.s file that was generated. Run this new binary and report on your findings.

Lab 1.2 Exploring different data formats

10. Write a program (*convert*), which first reads in two integers via scanf. It should print them out in decimal, hexadecimal and octal formats.
    Describe how you think the printf( ) function convert the integer value into the desired base (hexadecimal and octal) base.

Lab 1.3 Logical Operations

11. Create a new program *logicalOp* (hint: start out with a copy of *convert).* It should generate and print bitwise AND, OR and XOR values for your two integer input values. Make sure that these are printed out.

12. Finally, run your program, inputting negative numbers (try only one negative integer, then trying setting both to negative integers). What happens and why? Try to explain what is happening.

Lab 1.4 Two's Complement

13. Create a program (*twosComplement*) that asks for 1 integer value (decimal) and prints out the one's complement and two's complement representation of the input value in hexadecimal format.

Lab 1.5 Explore Pointers

Copy the following program, compile it and run it. Try to explain the results by adding comments to every line in the code.

```
#include <stdio.h>
int main(void)
{
    char ch = 'T';
    char *chptr = &ch;
    char name[6];

    int a = 1000;
    int *intptr = &a;

    float fnumber = 1.20000;
    float *fptr = &fnumber;

    char *ptr = "My dog has fleas!";

    printf("\n [%c],[%d],[%f],[%c],[%s]\n", *chptr, *intptr,
*fptr, *ptr, ptr);

    chptr = ptr;

    printf("\n [%c],[%s]\n", *chptr, chptr);

    name[0] = 75;
    name[1] = 97;
    name[2] = 0x65;
    name[3] = 0154;
    name[4] = 105;
    name[5] = 0;

    printf("\n [%s]\n", name);

    return 0;
}
```

Laboratory Report

Your report should be developed on a word processor (e.g., OpenOffice or LaTeX), and should include graphics when trying to present a large amount of data. Include the output of compiling and running your programs. Upload the lab report on blackboard. Attach a listing of each program that you wrote in the appendix to your lab report.