Engineering 304
Lab #3: NIOS II Assembly Language – Functions and the Stack

Purpose
The purpose of this lab is to learn how functions are called, how registers are used, and how the stack is used to protect registers that are used.

Overview
This lab will focus on the use of function calls which require the use of the stack and the passing of arguments and results. You will be combining assembly language and C language code together into a complete program. You will be provided a main routine that calls a function that you will write. The function that is called will be written as a recursive function which will then call itself.

Overall Requirements
Throughout this lab, you will be developing a program that will calculate factorials. The main() function will be provided for you and you will write the function that calculates factorials. The requirements listed below apply to both main() and factorial() functions.

- The main function shall generate n! (n factorial) for 0 <= n <= 12 by calling factorial() 13 times,
- All factorial calculations are done recursively in the factorial function and not in the main function,
- The main routine shall store each factorial value in an array (global variable) in memory,
- Subroutines/functions are **not** allowed to access the global array, any other global variables, or any register used by the main function (except when explicitly passing arguments and results),
- The main function shall only supply the argument "n" to the factorial function,
- The factorial function shall only return the value of "n!" to the main routine after completing the recursive calculation,
- All assembly language functions shall properly use registers according to the conventions outlined in the CPU reference manual and shall use <u>at least one</u> of the registers R8-R23 that must be saved,
- All functions shall properly use the stack according to the conventions outlined in the CPU reference manual, and
- The NIOS cpu shall be based on the provided *.sof file, *.jdi file, and *.qsys file.

Program Specifications
For this one-week lab, you will build and test a version that has the main function written in C (this is provided for you) and the factorial function written in assembly (a template file called "FunctionTemplate.s" is provided for you). Each function should be saved in a separate file (named main.c and factorial.s). Mixed C/assembly programs must be compiled by the C compiler, not the assembler, and so your debugger project must be configured as such. The label "factorial:" in your factorial function must be at the beginning of the function and must be made global (".global factorial"). You do not need to ".include" other files in your assembly file when compiling assembly with the C compiler.

Recommended Approach
It is recommended that you write out a detailed pseudo-code for the recursive factorial function. You will want to keep each statement in the pseudo-code as simple as possible so that they can easily be translated into assembly instructions. When implementing the assembly code, first choose registers, then implement the algorithm in assembly, and finally add instructions to handle the pushing and popping for the stack.

Hand In
Create, print, and turn in a Microsoft Word document that includes:

- A listing of your assembly code factorial.s. (same formatting requirements as labs 1 & 2).

- A screenshot showing the factorial values in memory (use the memory tab) after the program has completed. Be sure to include the addresses where the values are found in memory in the screenshot.
- A table listing your register usage (what registers were used and what they were used for) for the assembly language function that you wrote.
- A table listing the memory and/or register locations for each global and local variable used in the C code.
    - Determine this by analysis and debugging of the program. Each C-code global and local variable should reside in either memory or a register. If the variable is in memory, provide the memory address and size of the variable in bytes. If the variable is in a register, provide the register number.