SER 250 Project 3: UART String Conversion

Learning Objectives:

- Create modular code and interface with unfamiliar modularized code

The Task

In this project, you will be writing a program that converts a string to an integer (similar to Java's *Integer.parseInt()* method or Python's *int()* function). You program will receive ASCII strings from the UART, one character at a time. It should convert each string of characters into an integer value that is passed to a provided print function. Strings will be terminated using a semicolon (;) character and will contain one or more characters in addition to the semicolon. Your program should be able to handle multiple strings concatenated together and treat each string ending with a ';' as a separate input. When processing strings, it should detect any invalid characters within a string. Invalid characters are any characters other than '0' through '9' and ';'. If an invalid character is detected then the rest of the string (i.e. all characters leading up to and including the next ';') should be received by the UART, but ignored. Your program should then use the print function to output an error message and then continue processing the next string as a new string.

Print Function

A skeleton PLP project file is available to download on Blackboard. The PLP project includes a second *ASM* file titled, *project3_print.asm*. This *ASM* file contains the print function used in this project. PLPTool concatenates all *ASM* files within a PLP project into a single location in memory (unless additional *.org* statements have been added to specify different location for code). No changes to *project3_print.asm* should be made.

When called, the print function will send the value currently in register *$a0* over the UART to the PLPTool simulated UART device. Register *$a1* is used as an invalid character flag for the print function. If ***$a1* register contains a non-zero value,** the print function will display an **invalid character message** instead of the value in register *$a0*. The print function is called using the following instruction:

```
call project3_print
```

To use the print function, your PLP program needs to initialize the stack pointer (*$sp*) before performing the function call (or any other operations involving the stack pointer). For this reason, the skeleton project file includes an initialization that sets the stack pointer to `0x10fffffc` (the last address of RAM). This initialization only needs to be done once at the start of the program.

Deliverables:

1. Take the Project 3 Pre Quiz on Blackboard (3 points)
2. Submit your program on Blackboard with the format: *lastname_project3.plp* (25 points)

   Note: Please do not zip the project file or include any additional files in your submission
3. Take the Project 3 Post Quiz on Blackboard (2 point)