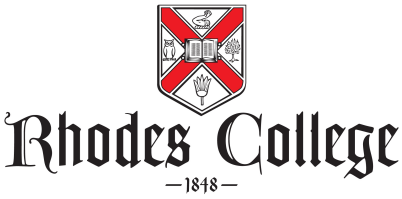
COMP 231
Lab 6



For this lab exercise, you will use the online NIOS II CPU simulator. Your program must work on the simulator system. **You only need to submit your assembly-language source file**.

Assembly Language Programming

This lab requires a functioning assembly-language implementation of the SSD lookup table from the last lab. Please contact me if you were unable to get the previous assignment working correctly.

For this assignment, you will implement a Nios II assembly language version of the following C program:

```
1   int fact(int n) {
2       if (n == 0) {
3           return 1;
4       else
5           return n * fact(n-1);
6   }
7
8   int main(void) {
9       int val, res;
10      val = read();       // read 8-bit value from switches
11      res = fact(val);
12      printHex(res);        // print 8-bit value to SSDs
13      return 0;
14  }
```

You should write this program using the standard Nios calling conventions and stack frames. Your program should begin in the main() function, read a value from the switches, call `fact()` , and print the values on the SSDs. You will also have to create functions `read()` and `print()` that read from the PIO switches and write to the SSD display. You may write the values to the LEDs if you wish, but it is not required.

In your code you must initialize the stack pointer before you can run your program. To do this, add the following code near the beginning of your program:

```
__default_stacksize=1024

_start:
         movia sp, STACK
```

And the following in your `.data` section:

```
.data
...
.skip __default_stacksize
STACK:
.end
```

Programming with Functions

To complete this lab, you will need to implement the following instructions using the proper Nios calling conventions and stack frames:

- **int main(void)** – This is the `main()` function in the program listed above. You should use the label `_start:` for this function. The main function calls three functions that you must implement: `read()`, `fact()`, and `print()`. You must follow the calling convention rules that apply for `main()`.
- **int read(void)** – This function reads an 8-bit value from the switches and returns it to the main function. This function should be very short.
- **int fact(int n)** – The C code for this function is listed above. Your function must adhere to calling conventions. Since it is a recursive function, you *must* save the value of n to the stack in some fashion.
- **void printHex(int val)** – This function takes a 32-bit value and displays the lower 16-bits in hexadecimal on the bank of four SSDDs (We can only use 4 of the 8 SSDDs with the simulator, so are limited to displaying max 16-bit values). This function relies on a helper function, `lookup()`, which provides the matching value from the lookup table array (`lut`) that was used in lab5.

  This function should loop over each 4-bit (nybble) value from the input, lookup the corresponding SSDD pattern, and place the pattern into the correct place within a register used for holding all 4 SSDD values in it. When you have decoded all four 4-bit input values into a single 32-bit output register (each SSDD displays the lower 7-bits in each byte of the output register).

  For example, if the lookup values for each SSD were `0xAA`, `0xBB`, `0xCC`, and `0xDD`, then `printHex()` should place all of the values into a single output register: `0xAABBCCDD` and write it out to the hex display address.
- **int lookup(int val)** – This function takes a single 4-bit hexadecimal value and returns the appropriate bit pattern to display on the SSDD. For example, calling `lookup(8)` should return `0xff`.

Project Setup

You should write all of your functions in the file `lab6.s`. When you have your program working, submit your complete assembly language program to Moodle.

NOTE: You must comment your code. Submitting a correct, but uncommented solution will result in a 25% penalty deduction.