

**Instructions:** The assignment is due on the date shown above. Tips to remember: give the assignments to your TA in section, remember your name, section number, TA name, and assignment number (5 points). Also, make sure your assignment is neat, stapled, and is entirely **your own work**.

1. Write an assembly program for the LC-3 that reads a number  $n$  from the console and computes the  $n$ th fibonacci number. Recall that the fibonacci sequence  $f(n)$  is defined as:

$$\begin{aligned}f(0) &= 0 \\f(1) &= 1 \\f(n) &= f(n-1) + f(n-2)\end{aligned}$$

Your program must be recursive, rather than iterative. This means that you must start with  $f(n)$  and within it, call  $f(n-1)$  and  $f(n-2)$ , and so on. Use the downloadable file fib-8.asm as a template. You may assume that the memory location labeled INPUT holds the input number  $n$ , and you should store the output to the memory location labeled OUTPUT before calling HALT. You should use R6 as your stack pointer and be sure to load and boot the los before loading and running your program (so that it can set up R6 for you):

- (a) Draw a map of the activation record for your fibonacci call - make sure to include all of the necessary components.
- (b) Draw a diagram of the contents of the runtime stack after the following sequence of calls and returns, which represents a partial trace the execution of an initial call to fib(4):

```
call fib(4)
call fib(3)
call fib(2)
call fib(1)
return fib(1)
call fib(0)
return fib(0)
return fib(2)
call fib(1)
```

- (c) What's the largest number  $n$  for which your program can compute the  $n$ th fibonacci number? Why?
  - (d) At what number  $n$  would your program go off the rails? Why?
  - (e) Turn in your program, fib-8.asm, both in hard copy and electronically using turnin.
2. Given the C function listed below - describe what the function does and draw a diagram of the activation record for the function (state any assumptions necessary). Note that 'a' corresponds to the ASCII numerical code for the character  $a$ .

```

int baz(int inchar) {
    int result_char;
    if (inchar >= 'a' && inchar <= 'z')
        result_char = inchar - ('a' - 'A');
    else if (inchar >= 'A' && inchar <= 'Z')
        result_char = inchar;
    else
        result_char = NULL;
    return result_char;
}

```

3. Using the template provided in `vec8.c` (available on-line), write a function `vector_sum`, that takes as argument the length of a vector of integers and a pointer to the vector, and returns the sum of its elements. Compile this program using the LC-3 C compiler as follows:

```
/p/bin/lcc/lcc -o vec8 vec8.c
```

Make sure your code produces the correct result by loading and running the `vec8.obj` file on the LC-3 simulator (`lc3db`). Rename `vec8.asm` to `vec8_lcc.asm` and annotate the instructions for the function `lc3_vector_sum` with comments describing how the instructions implement your corresponding C function. Be prepared to explain how each instruction contributes to implementing the C function.

Hint: You might find it useful to step through the code using the simulator to figure out what the instructions are doing.

- (a) Turn in a printout of your C code and the annotated assembly code (only the function `lc3_vector_sum` need be shown in the printout of your assembly code). Don't forget to fill in the header with your name and section number.
- (b) Your program should compile using the LC-3 C compiler on Unix and run on the Unix version of the LC-3 simulator. Submit the files `vec8.c` and `vec8_lcc.asm` using the `turnin` command.
- (c) Describe the stack protocol used by the `lcc` compiler. Is it the same as what we discussed in class? If not, what are the differences?
- (d) You will notice from the above exercise that the translation of the C function to assembly by the compiler produces very inefficient code. Write a hand optimized version of the function directly in LC-3 assembly language using the template provided in `vec8_hand.asm`, and filling in the code for the function `vector_sum`. Your code should assume that the length of the vector is available in register `R0` and the address of the first element of the vector is in `R1`. You should put the result in `R2`.
- (e) Submit a printout of your file `vec8_hand.asm`. Don't forget to fill in the header with your name and section number.
- (f) Measure the difference in running time (from the cycle counter available in the `lc3db` simulator) between the `lcc` compiled and your hand-generated versions of the programs. Quantitatively, what accounts for the difference?
- (g) Ensure that your program works on the Linux version of the LC-3 simulator and submit the file `vec8_hand.asm`, using the `turnin` program.