ENSC 254 – HW5 - BigFib Assignment

Summer 2021 – Due July 14 – max 1 partner

Introduction

In the previous assignment, we created a subroutine called *bigAdd* that can do arbitrary-sized unsigned addition. In this assignment you will write a function/subroutine, either in C or ARM assembly, that will use *bigAdd* to calculate huge Fibonacci numbers.

Exercise – fibonacci function/subroutine

Your job this week is to write a function in C or a subroutine in ARM Assembly that will calculate Fibonacci numbers.

The prototype/signature for the function is:

```
int bigFib (int n, int maxSize, unsigned **bNP);
```

n is the Fibonacci number that you would like to calculate. (Our Fibonacci numbers start with $f_0 = 0$, $f_1 = 1$, etc., and we would like to calculate f_n.) We might try setting *n* to a larger than necessary value like 1,000,000.

maxSize is an integer that specifies how many valid words can be stored in the *bigNumN* numbers.

Parameters *n* and *maxSize* should be greater than or equal to 0.

The return value of the function is normally either *n* or, if overflow occurred before *n* was able to be accurately calculated, the largest Fibonacci number that could be accurately calculated before overflow occurred. The return value will be -1 if an error occurs, and in such a case *errno* should be set to *EINVAL* (as defined via errno.h) if an invalid value is passed as a parameter or *ENOMEM* if there is insufficient memory available in the heap for needed memory allocations.

bNP is a pointer to a pointer to a *bigNumN* number holding the Fibonacci number indicated by the return value. **bNP* is provided by the code calling *bigFib()*, and *bigFib()* will update **bNP* to point to the proper pointer unless an error is reported. The *bigNumN* number should be freed with *free*() after the subroutine returns and, depending on how big the heap is, before *bigFib* is called again.