

Program 6: FORTH decompile

CS 636/338—Due June 2005

1. Load the `gcd` program and execute:

```
HEX
' gcd .      ( You see the code field address of gcd. )
```

2. Load the `fibonacci` program from the file `lexdata.f` and execute:

```
HEX
' fib .      ( You see the code field address of fib. )
```

Compare the CFA information for these two functions. What is the CFA type tag for the type `FUNCTION`?

3. Now use the CFA of `fib` to dump the code and parameter fields of the `fib` function.
4. Print out the dump with triple spacing between lines; make sure that the columns still line up neatly. (I did this by transferring everything to a spreadsheet.)
5. Execute the following code and make a neat table of the results. (On a spreadsheet?)

```
HEX
' dup .      ( You see the code field address of swap. )
' 2dup .
' rot .
' CR .      ( Do this for every word used in fib. )
```

6. Now use your source version of `fib` and the addresses printed out above to decode the compiled version that you dumped. Identify the meaning of each byte that you can.
7. Now look at the parts that could not be decoded this way. Find the parts of this code that implement the control words such as “do”, “loop”, and “if”. Can you find the addresses?
8. Obviously, there must be some way to jump from the end of the loop back to the beginning, and to jump around one of the clauses of the if statement. How does this work on your system? Do you find any pointers? Do the best you can to explain it.