Fibonacci Fun

The Fibonacci sequence is one of the most famous sequences of numbers in mathematics. The first Fibonacci number is 0, the second Fibonacci number is 1, and in general, the next Fibonacci number in the sequence is the sum of the previous two. The first few numbers in the sequence are 0, 1, 1, 2, 3, 5, 8, 13, 21. The nth number in the sequence can be defined recursively like this:

```
def fib(n):
        if n==0 or n==1: return n
        else: return fib(n-1) + fib(n-2)
```

Your job is to write a Hmmm assembly language program named `fibonacci` that takes a single input from the user, call it `n`, and prints the first `n` Fibonacci numbers.

You will likely want to copy the contents of one register `rX` into another `rY` during the course of this problem. Take a look at the Hmmm reference page below — the command for copying `r1`into `r2` is `copy r2 r1`. Note that it reads right-to-left (as with assignment statements, e.g., `r2 = r1`).

For this problem, you may assume that the input `n` will always be >= 0. Here is are some sample inputs and outputs:

```
Enter number: 0
<no output>

Enter number: 1
0

Enter number: 2
0
1

Enter number: 10
0
1
1
2
3
5
8
13
21
34
```

Remember to have a comment for every line of code that you write. Also, test your program carefully, starting at `n == 0`.

Documentation for HMMM (Harvey Mudd Miniature Machine)

Quick reference: Table of Hmmm Instructions

Instruction	Description	Aliases
System instructions		
halt	Stop!	
read rX	Place user input in register rX	
write rX	Print contents of register rX	
nop	Do nothing	
Setting register data		
setn rX N	Set register rX equal to the integer N (-128 to +127)	
addn rX N	Add integer N (-128 to 127) to register rX	
copy rX rY	Set rX = rY	**mov**
Arithmetic		
add rX rY rZ	Set rX = rY + rZ	
sub rX rY rZ	Set rX = rY - rZ	
neg rX rY	Set rX = -rY	
mul rX rY rZ	Set rX = rY * rZ	
div rX rY rZ	Set rX = rY / rZ (integer division; no remainder)	
mod rX rY rZ	Set rX = rY % rZ (returns the remainder of integer division)	

Jumps!		
jumpn N	Set program counter to address N	
jumpr rX	Set program counter to address in rX	**jump**
jeqzn rX N	If rX == 0, then jump to line N	**jeqz**
jnezn rX N	If rX != 0, then jump to line N	**jnez**
jgtzn rX N	If rX > 0, then jump to line N	**jgtz**
jltzn rX N	If rX < 0, then jump to line N	**jltz**
calln rX N	Copy the next address into rX and then jump to mem. addr. N	**call**
Interacting with memory (RAM)		
loadn rX N	Load register rX with the contents of memory address N	
storen rX N	Store contents of register rX into memory address N	
loadr rX rY	Load register rX with data from the address location held in reg. rY	**loadi, load**
storer rX rY	Store contents of register rX into memory address held in reg. rY	**storei, store**