Hi! I'm really sorry for this WIP post. I cannot find a way to save a draft of it (without erasing its formatting) other than posting it as a private post. Please ignore this post for now, thanks!

Hi!

I found the additional course material on the FRACTRAN language (<https://www.cs.cmu.edu/~sutner/dmp/Conway-Fractran.pdf>) and find the language very interesting yet difficult to write in, so I want to make it easier by transpiling a higher-level language into it. I definitely need to do read more on FRACTRAN as a target language, so my currently attempt is probably *very* bad and unoptimized, but I'm curious on your ideas, and maybe we can find a better implementation!

Below are what I came up with so far.

Features

* None-recursive procedures (with pass-by-reference parameters)
* Transpiling from static single assignments
* Input/output a list of integers

Removing the "Primes" in FRACTRAN

I initially found the idea that "x in F(x) is an arbitrary number" very confusing, and the fact that inputs and outputs are encoded as a product of power of primes didn't help much. However the blog <https://esoteric.codes/blog/an-intro-to-fractran> reveals that primes in FRACTRAN can be seen as indices to an infinite-sized register tape, where each slot contains a natural number (denoted by the index's power). In other words, x represents a state consisting of potentially infinite registers. Moreover, each denominator in the FRACTRAN code acts as a gate for the operation at its nominator. Therefore I think we can abstract primes into register indices and forget about using prime numbers.

Moreover, the nominator can be a product of primes, so each instruction can alter the state freely (e.g., increment infinitely many states, etc.).

Also note that there is no distinction between a variable register and a register that acts as a flag for running a procedure (discussed later).

Reserved Registers

A few registers are reserved for certain procedures that relies on fixed register indices. We can denote procedures that relies on fixed indices "native functions". These are typically simple operations whose definitions are built into the transpiler and generated automatically. Here is a list of currently considered native procedures and their reserved registers ("params" denotes the indices of the procedure's parameters, "return" denotes the index of the procedure's return value, "local" denotes indices of local registers that must be cleared prior to invoking the procedure):

* add: params[3, 5], return[2]
* subtract: params[3, 5], return[2]
* mul: params[3, 5], return[2], local[7, 11, 13]
* divmod: params[3, 5], return[2, 7], local[11, 13, 17, 19]
* move: TODO
* copy: TODO

TODO: fixed procedure indices for the above functions

Procedure Indexing

My