Instruction set analysis

What instructions needs to be included in the 2DWPU instruction set?

## What’s needed (general)

Comparison instructions – most probably in Query category.

Another condtional query: Is Float NaN? +inf or –inf?

SWAP operations – how do they work? Passtrough? – might be composed instruction

## Instruction categories

* Query
  + Conditional and unconditional querying
    - Zero, Not Zero all user and other important registers
    - Integer Comparison (probably just using two of the registers), equal, smaller, bigger (if there’s space, possibly FPU comparison)
    - Important bits (like carry and such)
* Operation
  + Mathematical operations
    - Integer operations (ALU)
      * Addition, Subtraction, Multiplication (higher/lower, use special register for storing the higher value?), Integer division, Modulo (use special register and just have one operation? **NO**, this would make it more tedius to do just the module), Abs
      * Provide both bitwise and logical versions of logical instructions?
    - Floating point operations (FPU)
      * Addition, Subtraction, Multiplication, Division, Power, Sin, Cos, Tan, Log10, Log2, Ln, Abs, IntToFloat, FloatToInt, Exp, Max, Min, Frac, Sqrt, Root,
      * Some of these require only one argument, how will their two argument versions work then? It won’t make sense to perform this operation with two values, as only one value can be returned, also discarding the value won’t be probably the best solution (possibly store it in some register as a quick way of converting two values by one instruction?)
  + Logical operations
    - Bitwise and logical? AND, OR, NOT, XOR (each one twice, possibly XOR can be just once – bitwise), bit shifting and such
* Passthrough action
  + Moving data between registers, operating with the stack
  + If there’s space, setting and clearing important bits
  + Increment, decrement
  + RETI for interrupt processing - there’s a good usage of the < version, as it can be put on the beginning and once the algorithm is done and returns back to this one, the RETI action will be carried out
  + Setting and clearing user bits (four of them)
  + IMPORTANT
    - Need to create a set of instructions to control parallelism and also calling some program blocks with full privileges
    - It would be prudent if these instructions would be passtrough, as they can get executed for example when a portion of branc his entered and also when it’s left (exited), thus ensuring proper synchronization and no collision of resources
* Index
  + Not required, there are no specific operations, all possible instructions already described
* Extended (trough Index)
  + If there’s not enough space for enabling and disabling important bits, then extended instructions can be used, as there’s tons of space
* Instructions needed for parallel processing
  + Branch query – unconditional query that branches – splits the execution in two
  + Fork – doesn’t execute both parts at once
  + Make more space – It’s possible to get more space if Index-Register instructions are modified – there’s much less than 1024 registers, so the index key can be lowered and several more bits can be utilized
  + ??? Are instructions to wait for a specific branch even wanted? Shouldn’t the processor encourage writing programs that can be executed fully in parallel, without one branch depending on the other one? Thus, there will be no possibility for race conditions to occur. **But** how will be accessing same resource solved? Perhaps a style, where calculation is done first and only then data are written back should be encouraged