Save all transactions forever. Nodes will store lots of data, but fuck that.

Contracts are the main way of changing the state of the blockchain,

Design choice:

Transactions and contracts as separate things or transactions made as an implementation of a contract, but still with some measures to prevent double spending?

Local state of contract could be stored as extra bits appended to the contract code?

90 stake = 90 winrate, how?:
1 - (1-h)^s

h = hardness

s = relative stake

s1 + s2 = s

(1-h)^s1 * (1-h)^s2 = (1-h)^s

Gas:

Put contracts into block even if they fail, miner still gets gas

Option 1: Simplicity

Option 2: Compile GO to EVM and use their gas analysis, GO -> EVM compiler already exists,

Option 3: Own functional language to -> IELE - LLVM with gas analysis, would need to run it in LLVM

Proof of work:
What are bakers doing?

What contracts do they choose?

When do they start computing contracts?

Completely open, exact way of mining is up to us - pick a solution and argue why you chose it

Baker and verification requires different interaction with the execution environment, figure out a way to include this in your interface.

Finalization:

How to prevent redoing the same expensive computations on rollbacks?