There are probably many reasons not to do this, but I was just thinking about baking in the common proxy pattern as a potential opcode.

Executing the opcode changes the state entries for the given account X to another account Y, forwarding ether balances and internal state to Y. Code in Y would stay the same (you would thus deploy a new contract with “upgraded” code separately), but would get the state from X merged with Y making “upgrades” much easier to handle than the complex and error-prone proxy patterns currently employed. X would have it’s runtime deleted (optional).

You would have to keep the storage slots consistent (or bake in means to directly specify a storage slot configuration which a compiler can use to build it’s contract), so this is probably not as safe as doing a manual upgrade, but would be nice if handled safely.

I would put it near the “selfdestruct” opcode because it is similar in destructiveness and is basically the more general case where instead of moving the state you destroy it (while still forwarding the ether balance). Asset balances in other contracts would also be unaffected, but that’s a whole different issue.

Thoughts?