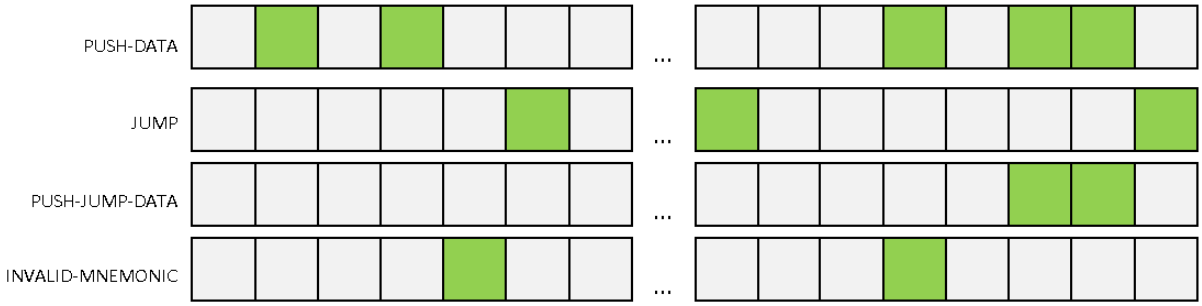
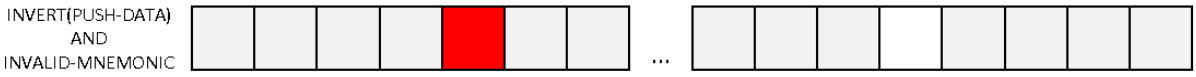
The goal of Ethertracer is to separate code and data of an Ethereum bytecode from each other. Given is the machine-code in hexadecimal notation.

We can preprocess the bytecode by searching and tagging mnemonics like Jump, Push, Jumpdest, etc. We are also able to find invalid mnemonics. In the given example 0x5c is an invalid mnemonic that isn't defined in the ethereum yellow paper https://ethereum.github.io/yellowpaper/paper.pdf



Based on the preprocessed lists, we can draw conclusions about the validity of segments. For example: If we encounter an invalid mnemonic that isn't part of a Push instruction, we can conclude that it must be a data segment.



Before we can tag a segment as data we have to split the bytecode in individual segments. Except from the contract start we define a segment as a sequence of instructions that begin with a Jumpdest instruction and end with [Jump, Stop, Selfdestruct, Return].

Given the clues that determine data segments and the segmented machine-code the final result is a list of Booleans, which determines code as true and data as false.

In total we use three techniques to find data segments. As mentioned in the example an invalid mnemonic that isn't part of a push instruction is one approach. Furthermore, we look for Push instructions that are directly followed by a Jump instruction. If the pushed data lies beyond the maximal address, we can assume that the segment is a data segment.

Finally, we check if the Jump-destinations are reached. Therefore, we collect all data that are pushed to the stack and compare them with the addresses of the jump-destinations. If there is no data on the stack that matches with a jump-destination we regard the jump-destination as unreachable and therefore the segment of the jump-destination as data.