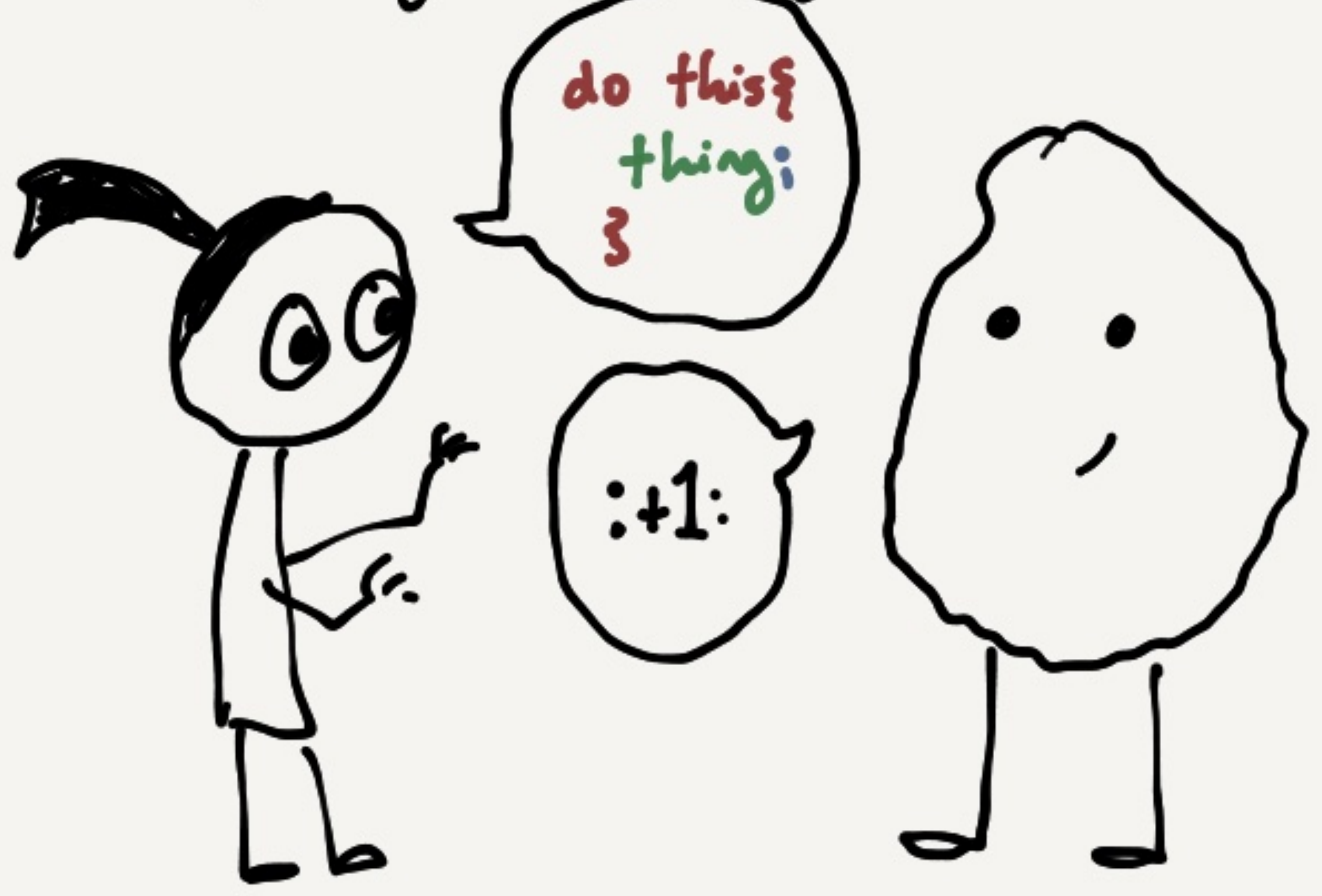
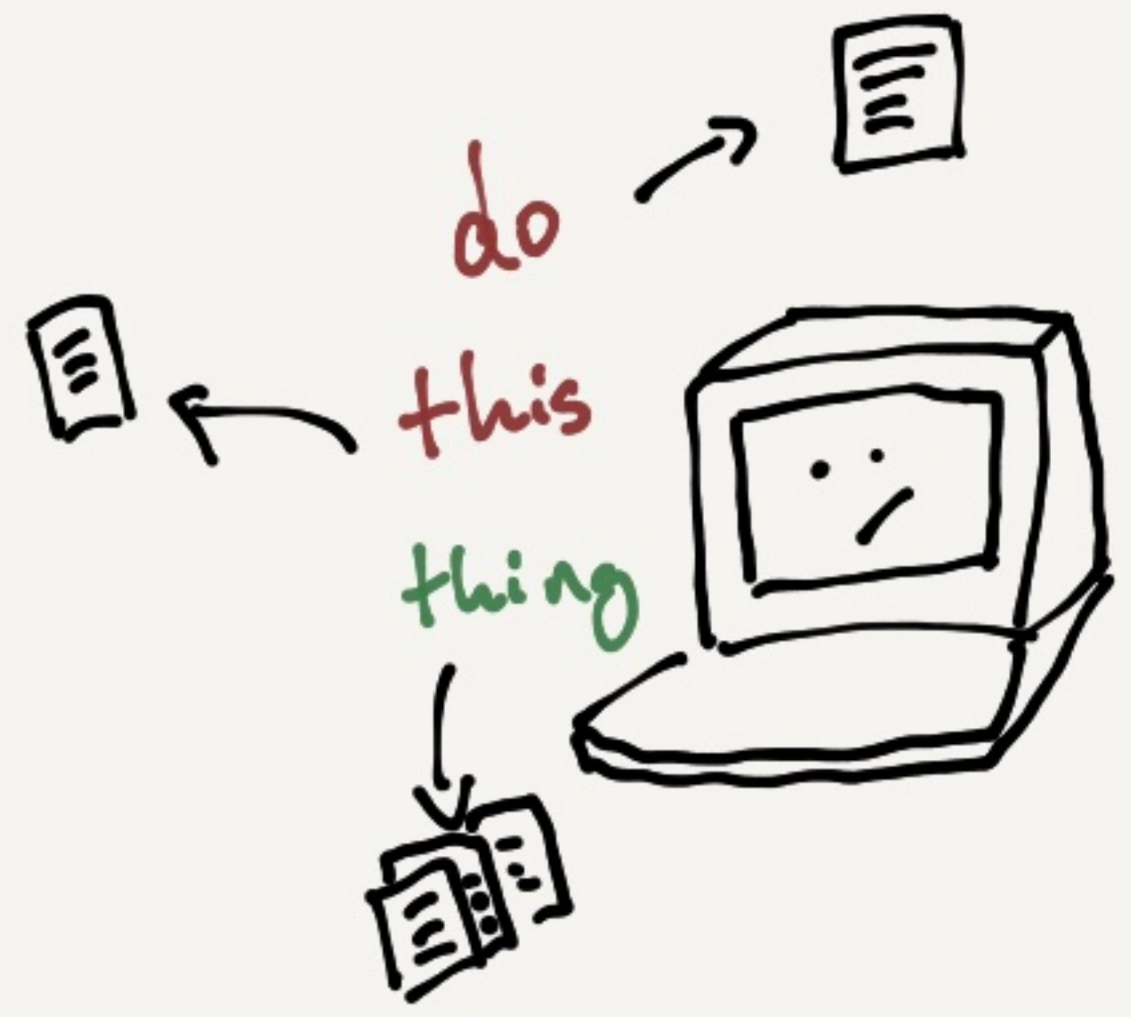
What is Bytecode?

High level programming languages (like JavaScript) help programmers

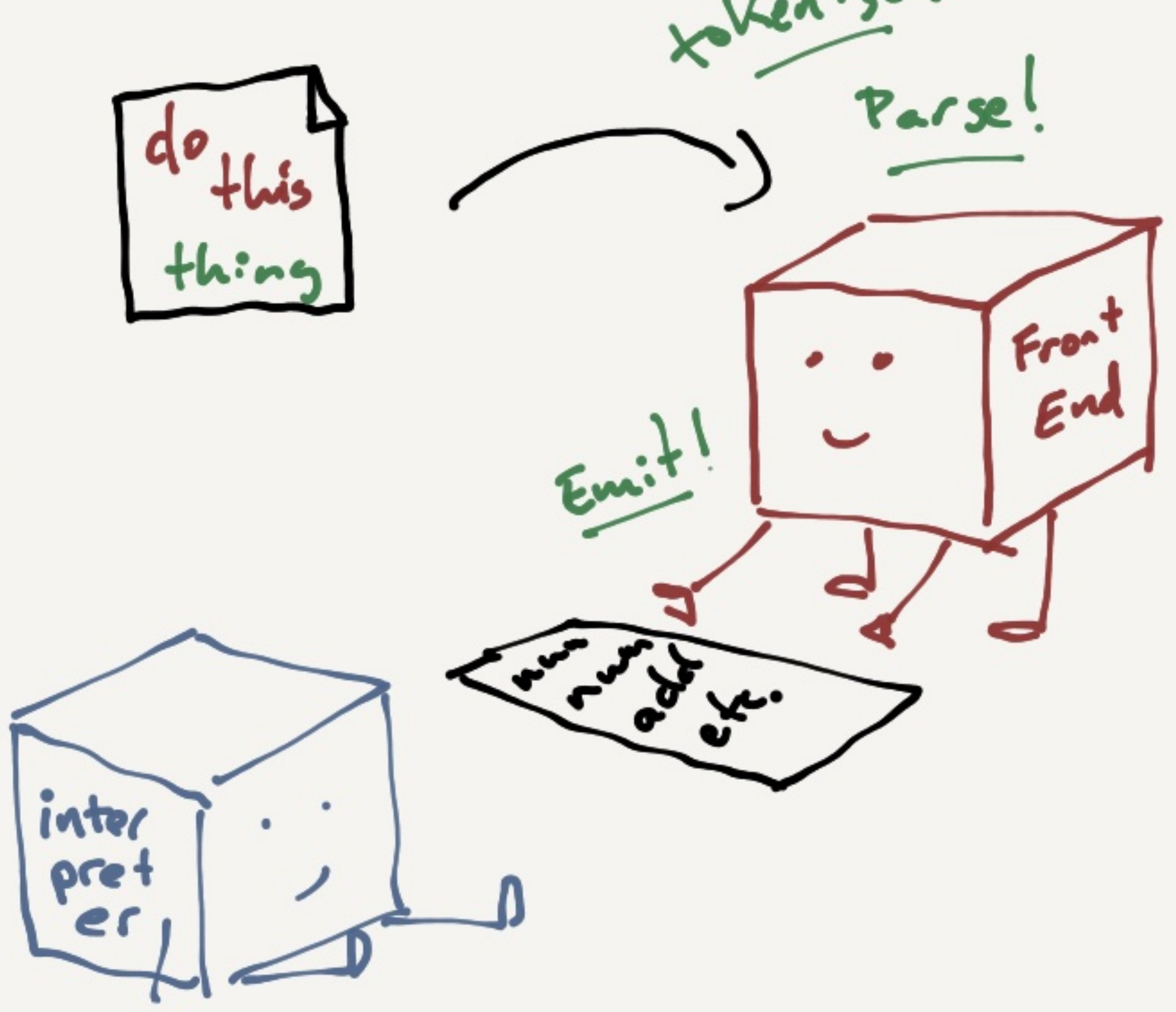


communicate the Intent of a set of instructions to other programmers.



But! these instructions are not something a computer can work with efficienty

So, interpreters and compilers translate sourcecode into an easier-to-process form. BYTECODE!

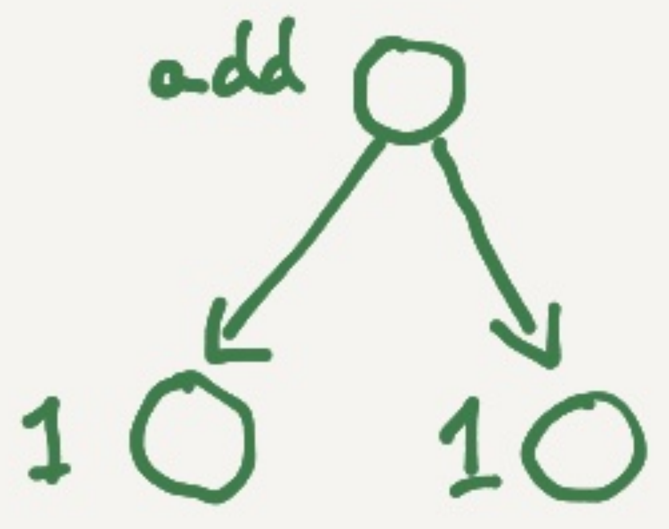


A concrete example from SpiderMonkey

Source code: add.js $\longrightarrow$ Abstract Syntax Tree

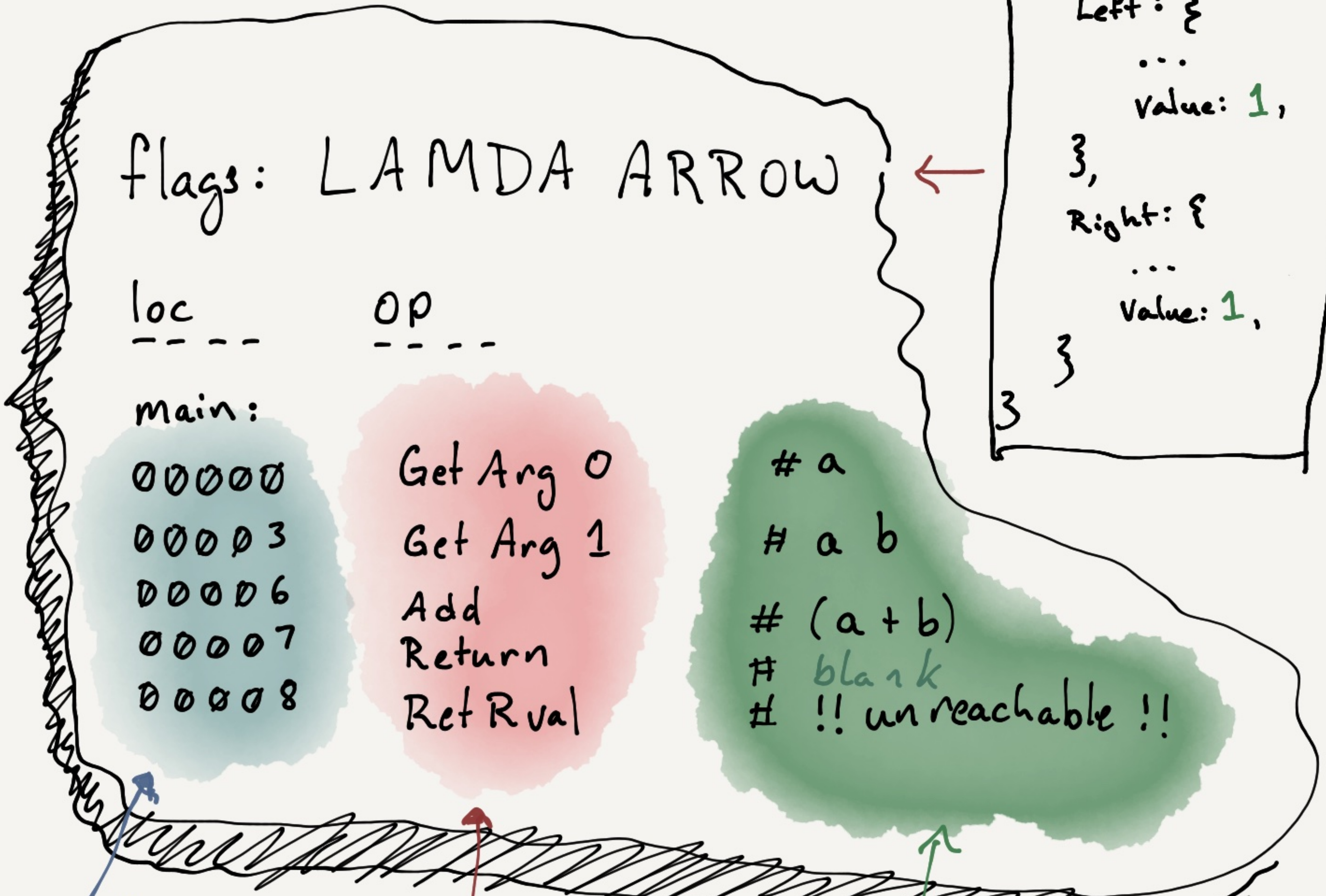
add.js
```
dis(  // helper disassembly function
  (a, b) => a + b
)
```

run with dist/bin/js add.js (see how to build)



or

Ast.json-ish
```
{ node: add
  Left: {
      ...
      Value: 1,
  },
  Right: {
      ...
      Value: 1,
  }
}
```



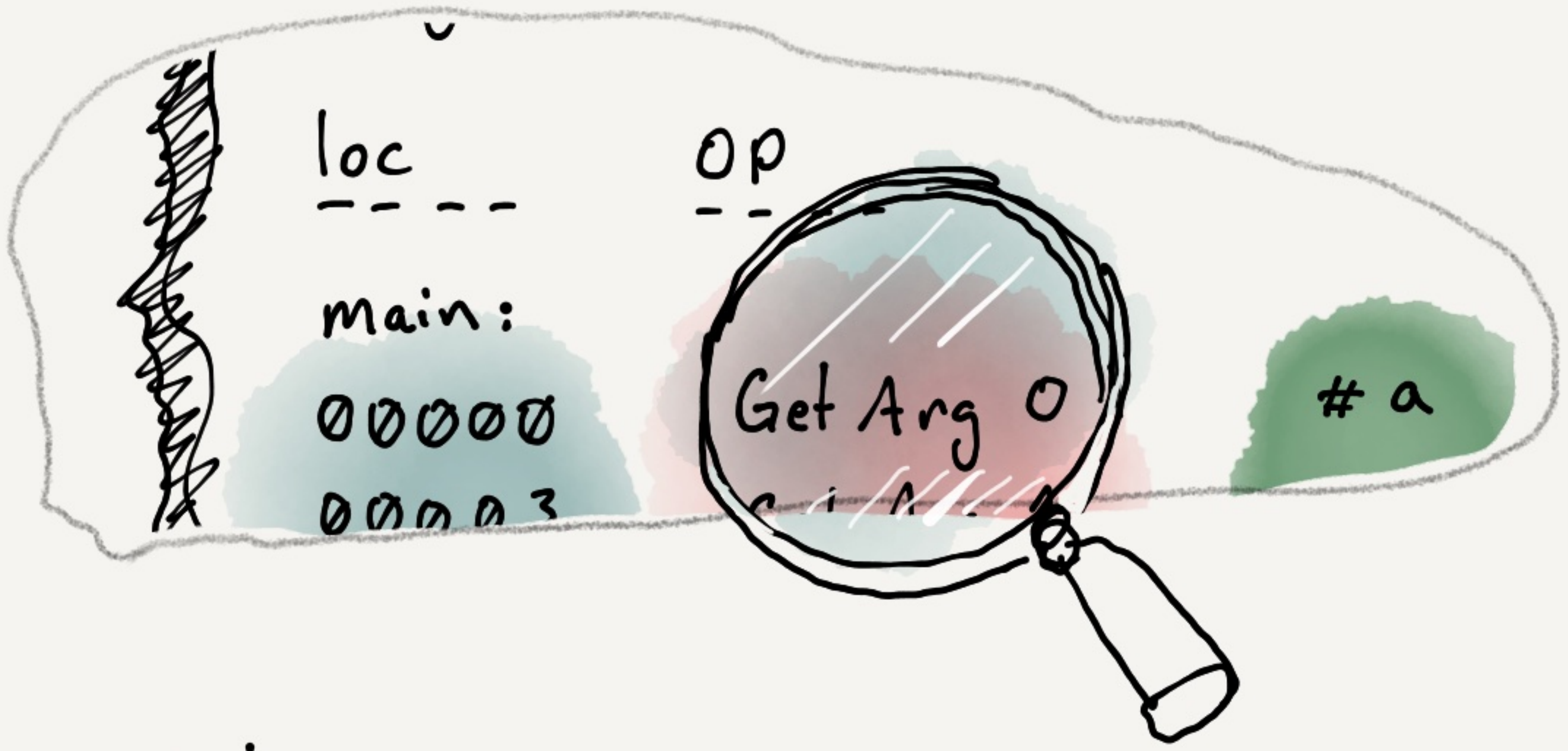
```
flags: LAMDA ARROW

loc      op
----     ----
main:
00000    GetArg 0     # a
00003    GetArg 1     # a b
00006    Add          # (a + b)
00007    Return       # blank
00008    RetRval      # !! unreachable !!
```

offsets

byte code operator name

The value stack

Zooming in on Get Arg



Note:

```
00000
00003 <- offset is the size!
```

use x/1x <address> in gdb or LLDB to get this

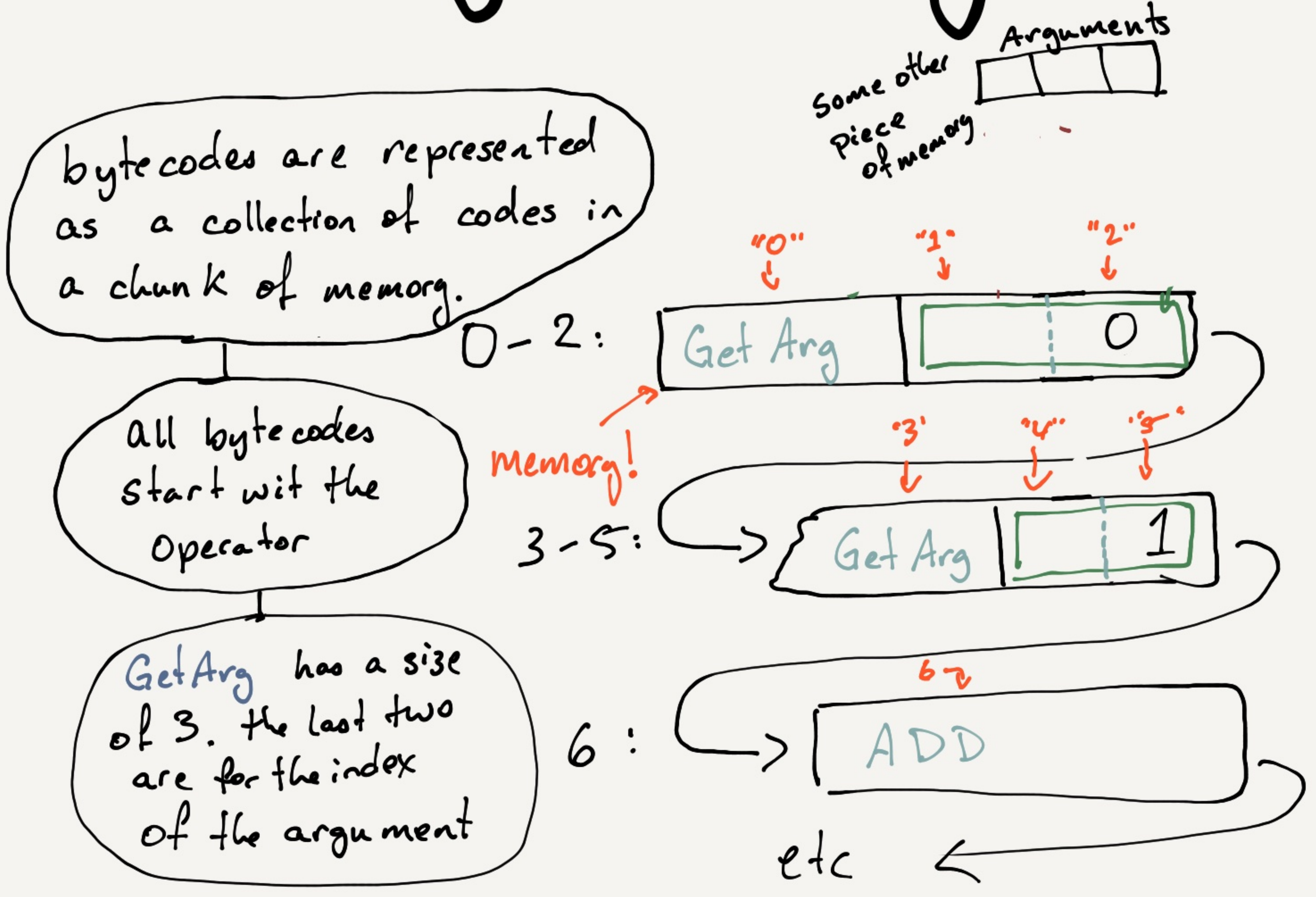
Byte code stack

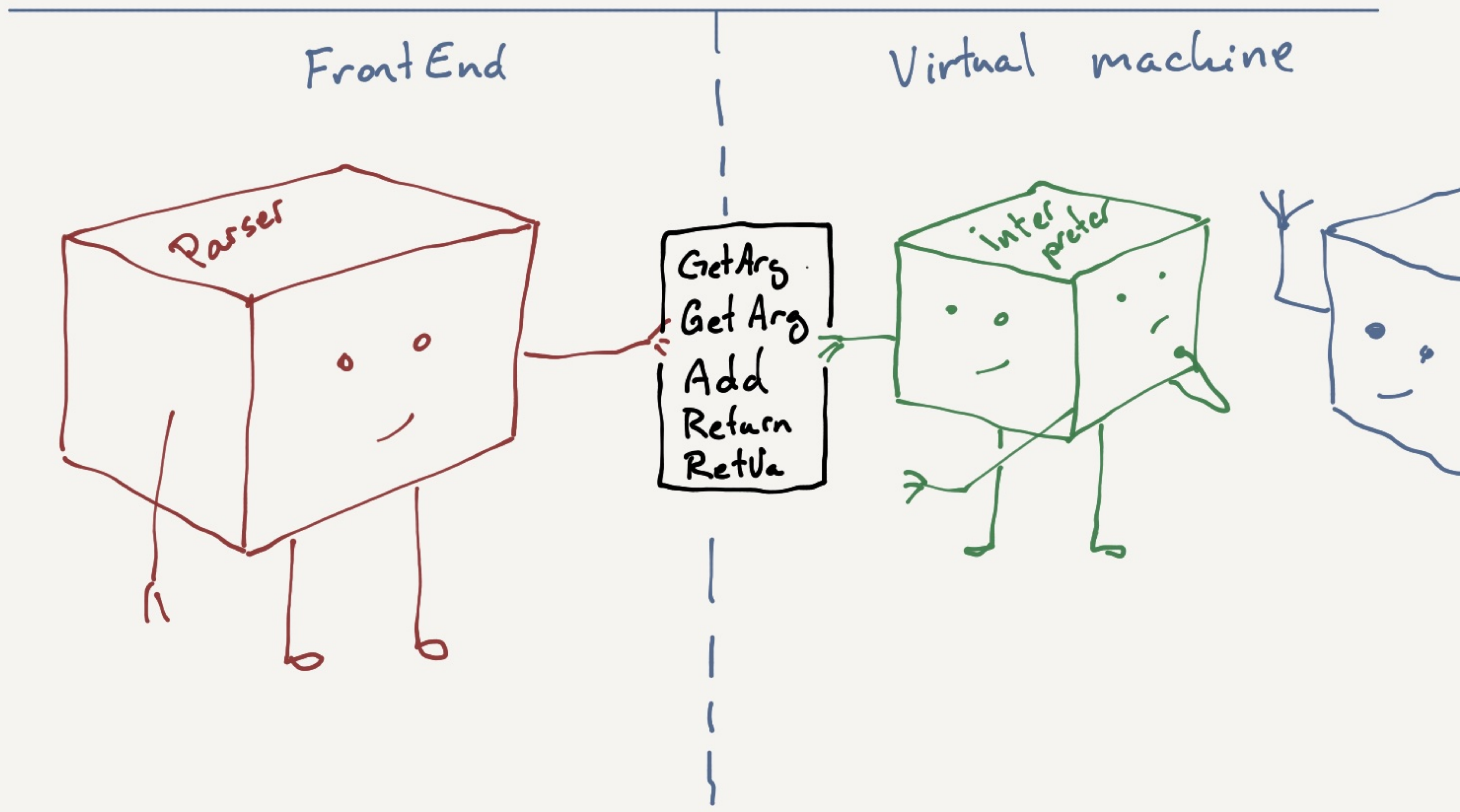
offset	Address (pc -> program counter)	value	
0:	0x000000001092af7e1	0xba	Get Arg
1:	" " 1092af7e2	0x00	
2:	" " 1092af7e3	0x00	
3:	" " 1092af7e4	0xba	Get Arg
4:	" " 1092af7e5	0x01	
5:	" " 1092af7e6	0x00	

0xba → representation of Get Arg operator

0x00 + 0x00 → u16 "0" represented as U8

Putting it together


Some other piece of memory
Arguments
bytecodes are represented as a collection of codes in a chunk of memory.
all bytecodes start wit the Operator
GetArg has a size of 3. the last two are for the index of the argument
memory!
"0"
"1"
"2"
0-2:
Get Arg
0
"3"
"4"
"5"
3-5:
Get Arg
1
6
6:
ADD
etc


Front End
Virtual machine
Parser
GetArg
Get Arg
Add
Return
RetVa
inter preter