Hack VM, part 2!

Branching instructions: label, goto, if-goto.

label name → (function_name $ name)
contains file name too!

goto name → @function_name$name
0;JMP

if-goto name

= pop from stack. if true, jump to name.

↳ JLT. — checks if 1st bit is 1.

Functions!

Everything is a function.

- function name m ← # of local vars.
- call name n ← # of arguments
- return

Bootstrap code!

SP = 256.

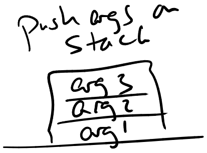
Call "Sys.init" 0

↳ more setup, call main.

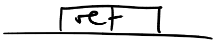
What are these supposed to do / how do they work @ the VM level?

- From POV of the caller:

① push args on stack

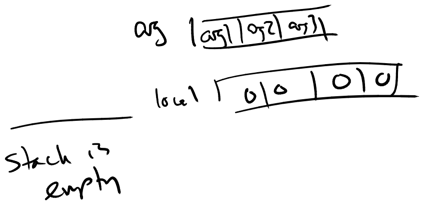


→ ② "call f 3" → ③ GO TO SLEEP → ④ return value is on stack.



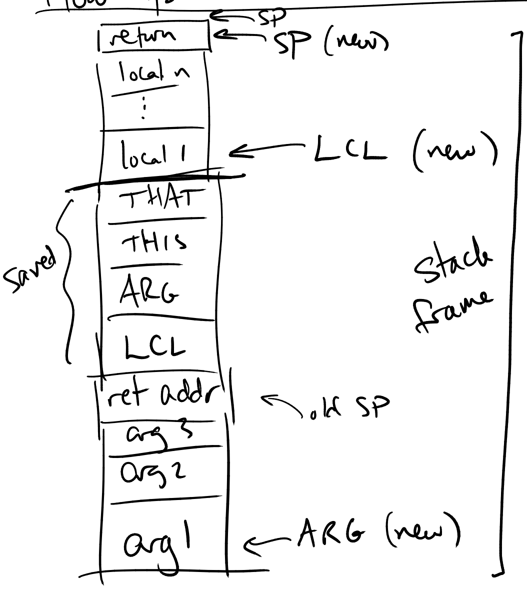
local, arg, this, that, static are unchanged.

- From POV of the callee (function which is called):



- stack empty
- arg segment contains args
- local segment filled w/ 0's
- Can do what we want w/ this, that, temp.
- When done, push return value on stack + call 'return'.

How to make this actually work? VM ⟶ Assembly. (pp. 155 – 161)



call name m

push return address @return5
push LCL, ARG, THIS, THAT
LCL = SP
ARG = SP − 5 − m
jump to filename.functionname.
(return5)

return

eg. R13.

frame = LCL

R14

ret = value @ (frame−5)

[*(frame − 5)]

value @ address

pop return value + store @ ARG.

SP = ARG + 1

THAT = value @ (frame − 1)
THIS = " @ (frame − 2)
... ARG, LCL.

jump to ret.