High-level

Virtual Machine ← next 2 weeks.

Assembly ↙ VM translator (projects 7 + 8)

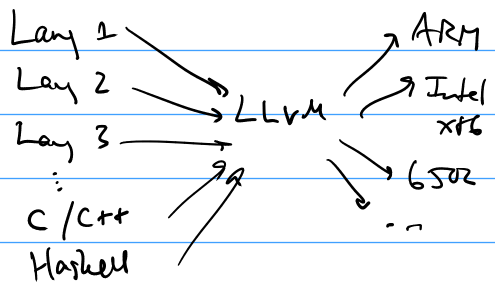
Machine

What is a virtual machine?

Basic idea: imaginary ("virtual") machine that is simulated on top of some underlying actual machine.

Common VMs

- Java Virtual Machine.
- Microsoft .Net — CLI (eg. C#)
- LLVM ("low-level virtual machine").



Hack VM

- Stack machine.
  - Has a stack of values
  - Arithmetic, logic etc. operate on the stack.
  - Saving/loading from memory always is to/from stack.
  - Functions are saved on the stack.
- Commands:
  - Memory access — today
  - Arithmetic + logic — Thurs.
  - Function calls } — next week
  - Branching }

Memory in the Hack VM

- stack
- various separate memory segments.

2 types of instructions:

- push segment index ⟹ push value at segment[index] onto the stack.
- pop segment index — reverse.

Segments

argument	— args passed to the current function.
local	— stores cur function's local variables.
static	— variables shared by all functions in the same .vm file.
constant	— fake, only for push — get constant value on stack.
temp	— only 8 values. scratch space for compiler.
this	— current object.
that	— arrays.
pointer	— only 2 values. pointer[0] = this, pointer[1] = that.

next week: this, that, pointer

Standard mapping VM → Hack machine.

Hex	Dec	Content
0x0 — 0xf	0 – 15	virtual registers.
0x10 — 0xff	16-255	static variables.
0x100 — 0x7ff	256-2047	stack.
0x800 — 0x3fff	2048-16383	heap. — stores objects + arrays.
0x4000 +	16384 +	Screen, kbd.

See below

virtual registers

0 — SP	stack pointer. Stores next free address on top of stack.
1 — LCL	points to cur location of local segment.
2 — ARG	" " " " " arg "
3 — THIS	" " " " " this "
4 — THAT	that
5–12	— temp segment. (scratch for compiler)
13–15	— whatever you want. (scratch for you).

x 256
y
z
← SP

Projects 7 – 8. Goal: translate VM code → assembly.

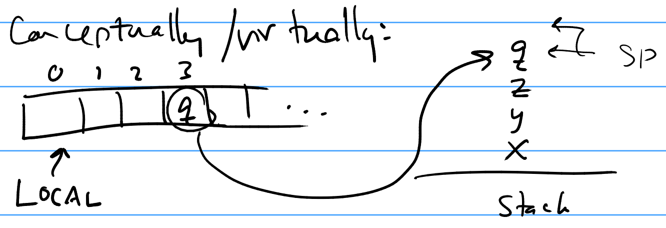
Similar to assembler. Linear, translates input code → output

Difference: – 1 VM instr → many assembly instructions
- Multiple input files.

Examples.

push local 3

Conceptually / virtually:



Concretely:

– Look up address stored in LOCAL	@LOCAL D=M
– Add 3 to it	@3 D=D+A

– Fetch the value stored at that address	A=D
	D=M

– Look up address stored in SP	@SP A=M
– Store the value at that address	M=D
– Increment SP.	@SP M=M+1

or A=D+A.