Lecture 22:

- MyPL Virtual Machine

Announcements:

- Proj Part 1 due
- HW-4 out
“Virtual Machines” (VMs) for PL Interpretation

Implements an “abstract (computing) machine”

- similar to computer hardware (but in software) …
- like a computer, consists of memory, instruction set, etc.
- instructions often similar to assembly (but often simpler and higher level)
- e.g., load, store, add, jump, etc.

A “bytecode” VM

- encodes instructions in binary as a sequence of bytes (e.g., .class files)
- e.g., ADD 3 4 might be encoded for the VM as 0 1 1 0 0 0 1 1 0 1 0 0
- "opcode" "3" "4"
- keeps programs smaller and less effort to “parse” input programs
MyPL VM for HW-5 and HW-6

- Based loosely on the JVM architecture (stack machine, stack frames)
- Via API calls instead of using bytecode encoding/decoding
- Takes some short cuts, tailored to MyPL
- Performs minimal error checking (except for runtime program errors)

(1) **Data Types/Values**
- Uses Python types to represent values and assumes programs are well typed
- Uses Python *None* value for representing MyPL *null* values

(2) **Abstract Stack Machine**
- Instead of registers, uses an “operand stack”

```
Initial stack
V. ...
PUSH 3
V. ...
3
PUSH 4
V. ...
4
ADD
V. ...
7
```

The VM components include:

- operand stack (see above)
- memory for storing local variables
- struct heap storage
- array heap storage
- function-call stack (stack of call “frames”)

... more later