Review: the Stack Abstraction

Stack in memory similar to stack on a desk.

Operations include:
- **PUSH—put something on top of the stack**
- **POP—take the top thing off of the stack**

A stack
- provides last-in, first-out (LIFO) semantics:*
- first thing popped is the last thing pushed

*As opposed to first-in, first-out (FIFO) semantics, as with the queue that we used with BFS.

Review: the Stack Abstraction in LC-3

In LC-3,
- we use R6 as a stack pointer, and
- PUSH/POP require two instructions each

Most ISAs
- **have a stack pointer register and**
- **include PUSH/POP instructions.**

The Stack at This Level is Not Checked

P&P talk about overflow/underflow checks.

That's fine when we reach C.

High-level languages (such as C) rely heavily on the stack provided by the ISA.

The stack provided by the ISA
- **is typically unchecked**,
- as checking overhead is too high, so
- don't make mistakes.

What Really Happens with Overflow/Underflow?

If a **stack overflows**…
- in LC-3/embedded processor/inside OS,* causes **silent data corruption**;
- in desktop/laptop/phone application, hardware detects, and OS causes **program to crash**.

If a stack underflows…
- silent data corruption is likely to happen first, and
- program may crash.

*For example, inside your OS in ECE391.

