I/O

- Ultimately, interaction w/ I/O devices involves reading & writing regs.
  (nowadays mem to)

- 2 mechanisms for transfer
  - Special I/O fnstres.
    e.g.:
    \[
    \begin{array}{c|c|c}
    \text{IN} & \text{DR} & \text{DEV} \# \\
    \hline
    \text{OUT} & \text{SR} & \text{DEV} \#
    \end{array}
    \]

- "Memory mapped I/O": Just use regular loads/stores, but from/to special addresses.

[WHY?]

1) Pin limitations
2) overhead count
3) Flexibility

- These addr's do not go off to DRAM chips, but to device regs.

- On LC-3
  \[
  x_{0000} \rightarrow x_{FFFF} = \text{MEM}
  \]
  \[
  x_{FEO0} \rightarrow x_{FFFF} = \text{MMIO ADDR}
  \]
  \[
  \leq 512 \text{ DEV REGS AT MOST!}
  \]

- In reality, things are much more flexible.
  PCI allows us to add/remove MMIO device & change where they live.
- On Linux, we can look at `/proc/iomem`

- There's a piece of software (BIOS) that detects all devices, and assigns addresses to them. Almost all systems use PCI bus for this (PCIe these days)

- With LC3 we can't plug in new devices.

- Must synchronize with I/O devices somehow (handshakes / READY / STATUS bits)
  (Synchronous, Asynchronous, Discussion)

- Polling vs. Interrupt-driven I/O
LC-3 KEYBOARD

2 REGS:
- KCBDR (DATA)
- KBSR (STATUS)

This is pretty common for simple I/O DEVS.

- KB H/W puts ASCII code in KBDR
  When user types key, sets ready bit in KBSR to 1

- CPU reads KBSR, if data avail.,
  can get code from KBDR.

- Reading automatically clears KBSR[15]

KBSR is @ xFE00
KCBDR is @ xFE02