Two Schemes Exist for Accessing I/O Registers

How does the processor access I/O registers?

How we really name I/O registers?
With bits!
(That’s really ALL we have, remember?)

So we have two options:
1. create a new space of names, or
2. use an existing space of names.

Option 1: Add I/O Instructions, Such as IN and OUT

Let’s think about the first option:
◦ a new space of names
◦ (names are bit patterns).

To access this space,
◦ we need to add opcodes
◦ that use the new names,
◦ for example, IN and OUT.

Few modern ISAs use this approach.
x64 (and x86) are the notable exceptions.

Option 2: Use Memory Addresses for I/O

For the second option, we can
◦ use part of the memory namespace
◦ and perform I/O with loads and stores.

This approach is called memory-mapped I/O.

Most modern ISAs use this approach.
So does the LC-3 ISA.
x64 (and x86) also support it.
Individual Names (Bit Patterns) Cannot be Shared

Note that
- **using** some “memory” **addresses for I/O**
- implies NOT using those addresses for memory.

This constraint can be an issue when memory addresses are few (16-bit or less).

In the LC-3 ISA,
- addresses xFE00-FFFF are used for I/O
- (1/128\textsuperscript{th} of memory).

---

Keyboard and Display are Memory-Mapped in LC-3

Keyboard and display registers are mapped as follows:*  
- xFE00 KBSR
- xFE02 KBDR
- xFE04 DSR
- xFE06 DDR

Let’s take a look at how I/O is implemented.

*Why every other address? Ask a student in 411 about LC-3b.

---

P&P Appendix C Describes I/O Memory Mapping

(Patt and Patel Figure C.3)

Memory control signals are delivered to “address control logic.”

---

Keyboard and Display are Memory-Mapped in LC-3

Based on MAR and these control signals, the **address control logic controls**:
- **memory enable** (chip select) signal actually delivered to the memory,
- **load control for DDR**, * and
- **INMUX select** lines, which determines whether memory, KBSR, KBDR, or DSR writes the load result to MDR.

*And for KBSR and DSR, but we’ll explain why later.
Example: Reading the KBSR

**read KBSR:**
- MAR = $\text{xFE00}$, R.W = read
- MIO.EN = 1

Example: Writing the DDR

**write DDR:**
- MAR = $\text{xFE06}$, R.W = write
- MIO.EN = 1