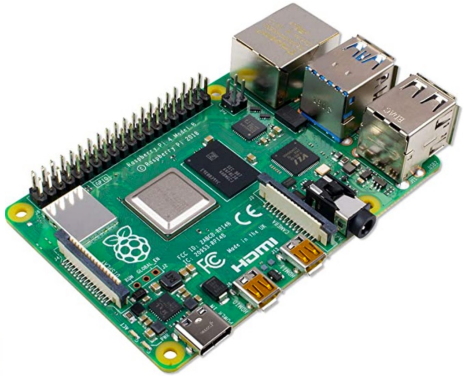
ECE 220 Computer Systems & Programming

Yih-Chun Hu
adapted from material by Profs. Yuting Chen, Sanjay Patel, Volodymyr Kindratenko

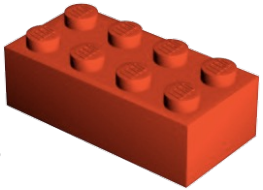


Ways to Support I/O

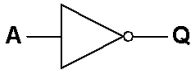
- Port-Mapped I/O: separate address space, separate instructions to access I/O devices
  - Example: x86 IN, OUT instructions
- Memory-Mapped I/O: map I/O devices to regular memory, access with normal memory instructions

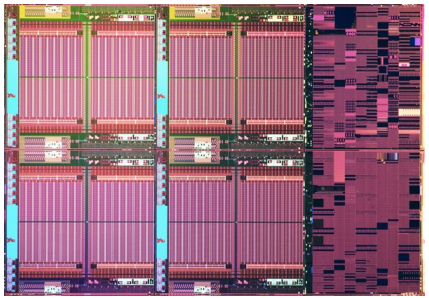
Key Concept: Abstraction

Create building blocks that are tightly specified.
Abstract way their details to a simple interface.
And then use them to build more
complex things.



Then optimize the building blocks like crazy…

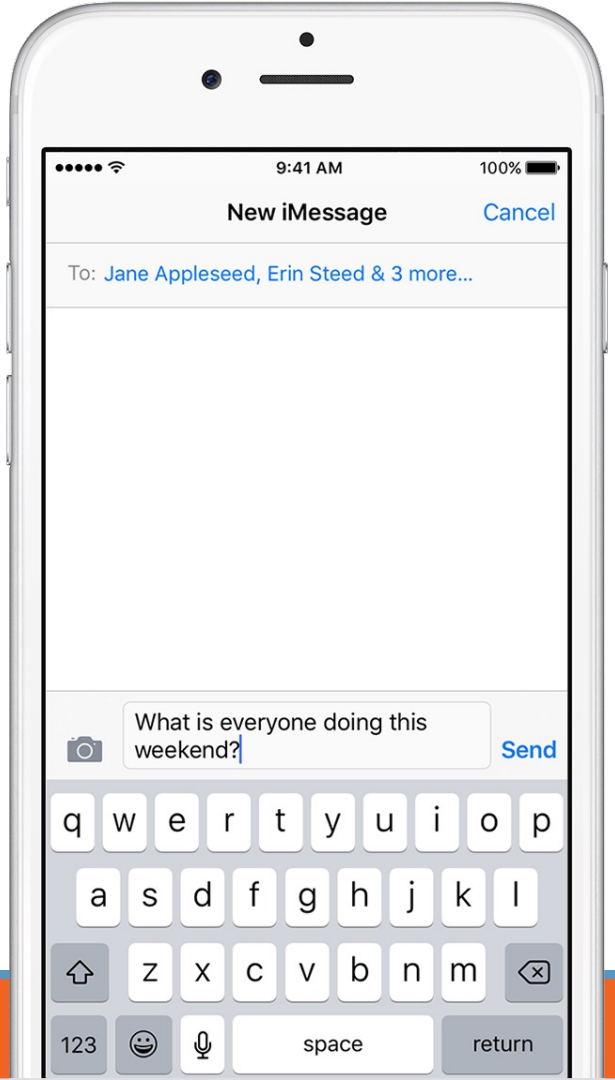


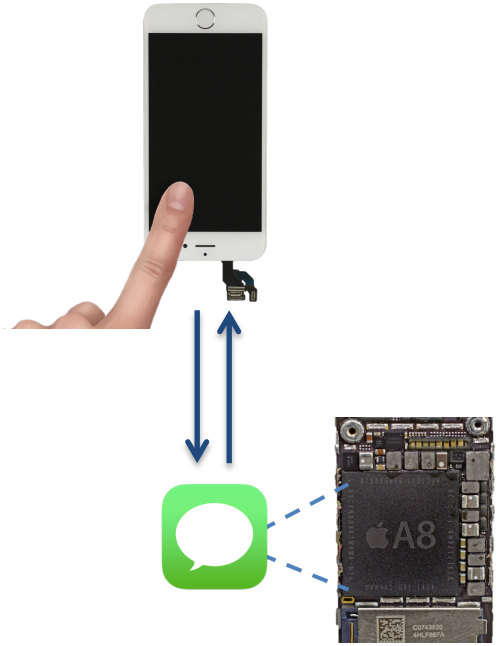


I/O is for interfacing between the physical world and digital world.

So what is the world's most commonly used digital interface?

….at least between a human and a digital system.





LC3 Memory: Memory mapped device registers

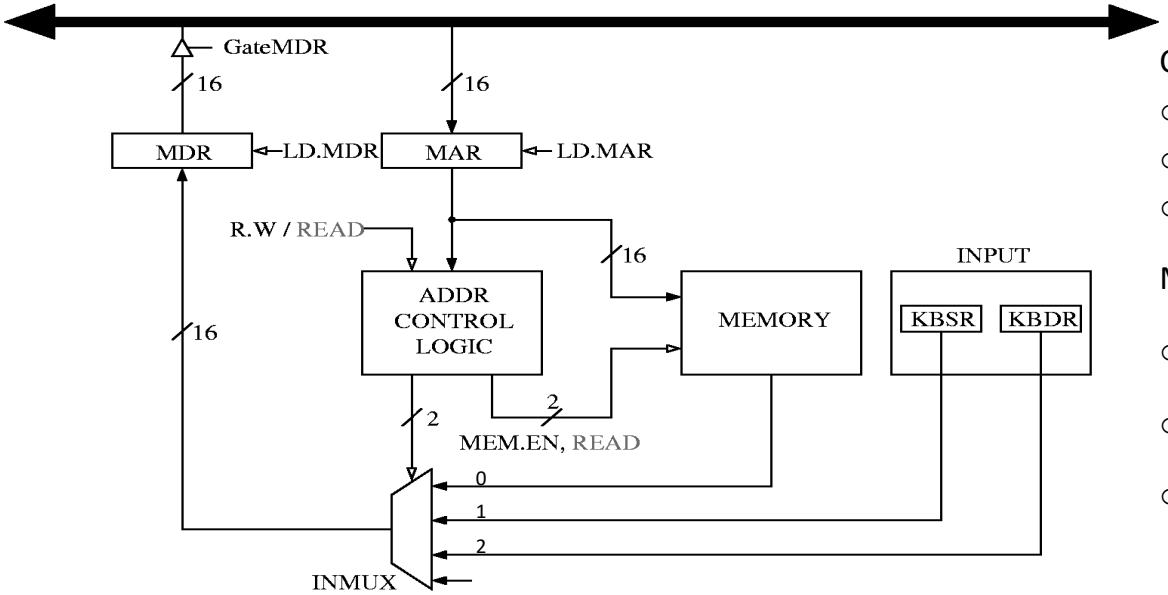
Address	Contents	Comments
x0000		; system space
…		
x3000		; user space
		; programs
		; and data
…		
xFE00	KBSR	; Device register
xFE02	KBDR	
xFE04	DSR	
xFE06	DDR	
…		
xFFFF		

These are the memory addresses to which the device registers (KBDR, etc.) are mapped

But the device registers physically are separate from the memory.

Memory-mapping device registers is a very common way to design interfaces for computing systems

Circuit for memory mapped Input



Conventional memory access: LD DR, addr

- MAR←addr
- MDR←MEM[MAR]
- DR←MDR

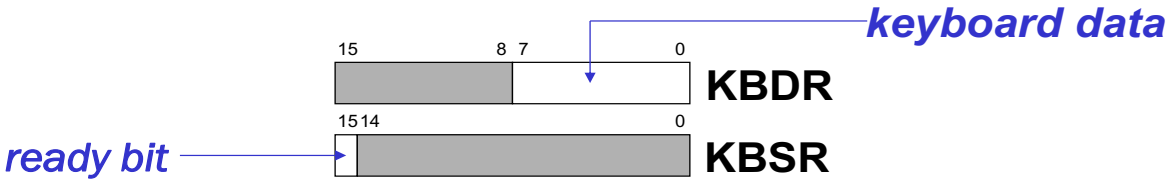
Memory-mapped input access: LD DR, xFE02

- MAR←xFE02
- MDR←KBDR
- DR←MDR

Basics of Interface Design

- Producer of data (finger at touchscreen) is working much much more slowly than consumer of that data (messaging app)
- We need to account for *asynchronous* operation
- We will use a simple consumer/producer handshake

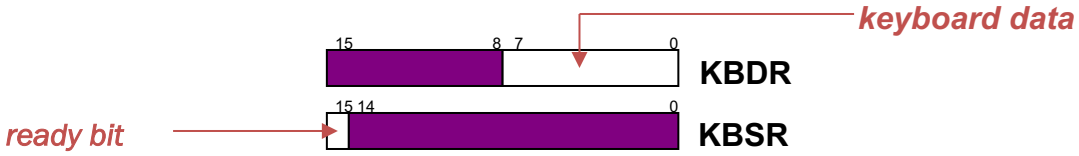
Handshaking using KBDR and KBSR



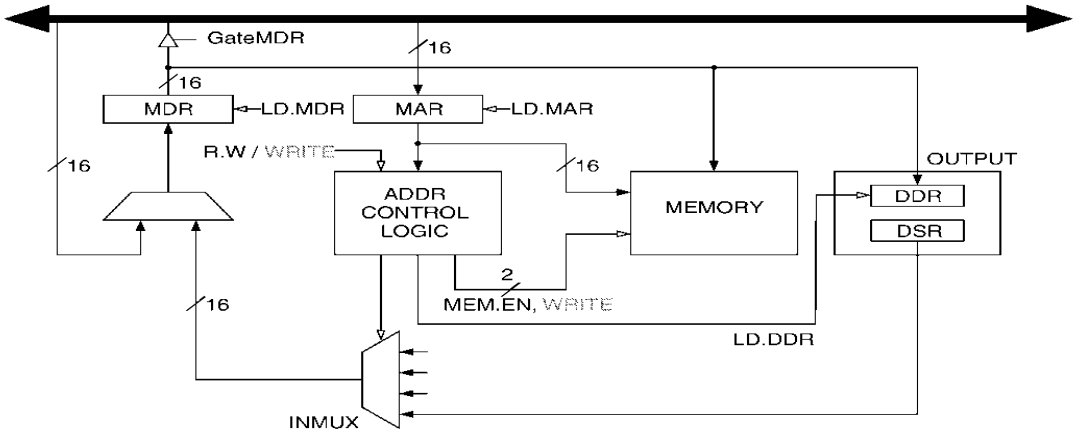
- When a char is typed by user in the keyboard
  - Its ASCII code is placed in KBDR[0:7]
  - KBSR[15] is set to 1 (ready bit)
  - Keyboard is disabled
- When KBDR is read by CPU
  - KBSR[15] is set to 0
  - Keyboard is enabled

This is part of the keyboard hardware.

Reading Input the right way



Circuit for memory mapped output



Conventional write: ST SR, addr

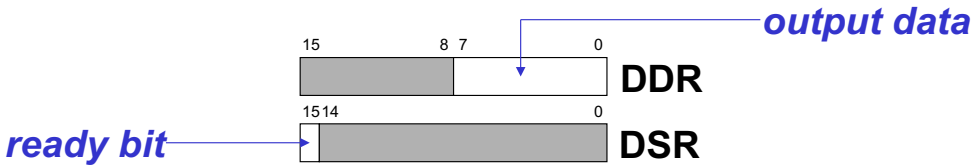
- MAR←addr
- MDR←SR
- Mem[MAR]←MDR

Memory-mapped input access: ST SR, xFE06

- MAR←xFE06
- MDR←SR
- DDR←MDR

Problem of asynchrony, again!

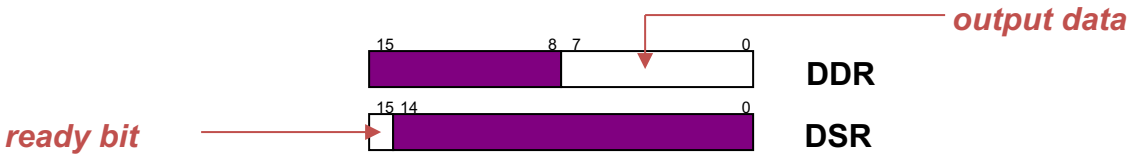
Handshaking using DDR and DSR



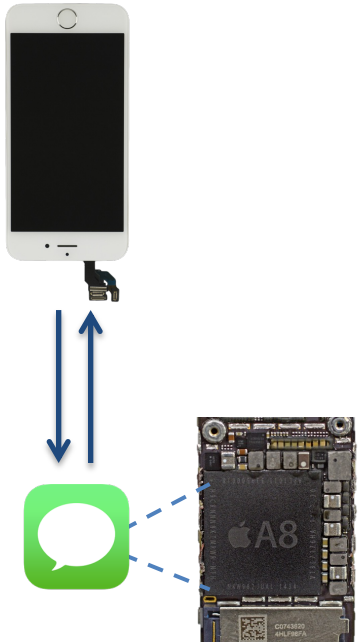
- When display is ready for another char
  - DSR[15] is set to 1 (**ready bit**)
- When new char is written to DDR
  - DSR[15] is set to 0 and DDR[7:0] is displayed
  - Any other chars written to DDR are ignored

This is part of the display hardware.

Sending Character to Display



But wait…



- What's the system doing while it waits for me to touch the next key?

  [hint: `BRzp START`…]

- If it takes me 0.5 sec, that's about 1 billion operations

- How can we do something useful?