These notes were written several days after the fact, so they're more of a synopsis of what we covered that day.

1. MMIO
2. Fibonacci, an example of a recursive program

1 Memory-mapped Input and Output (MMIO)

There are many ways to get our processors to interact with the physical world (although they'd be an interesting intellectual exercise, they'd be pretty much useless for everything we do with them today if we couldn't give them input besides the program itself, and couldn't get output).

Here are a few options:

Have dedicated I/O instructions

- believe some early instruction sets used this method; modern processors don't because many systems with different hardware all use the same instruction set, and hardware/hardware interfaces change many, many times during the lifetime of an instruction set. If you make an architectural decision, you're stuck with it for decades. **Note:** This is my conjecture for its lack of popularity.

Have registers connected to the physical world

- believe (although I may be very wrong) that the ATmega328 processor in the Arduino Uno (AVR instruction set) uses this approach. See next page.
After an hour of reading the AVR instruction set manual (one of the commonest microcontroller architectures—what Arduino uses, for example) and exploring Wikipedia, I realized how little I knew about input/output, and how difficult it'd be to summarize what I did figure out. It's really cool, but a bit outside of the scope of 004. If you want to talk about it one-on-one or few-on-one, I'd love to, for the recitation notes, though, we're going to stick to MM10.

Have "Memory" connected to the physical world

This is MM10—memory-mapped input/output. All that "memory-mapped" means is we get input and write/send output by using the regular LW and SW instructions we're used to.

Imagine that the processor is your house (at "1111 Processor Lane") and main memory is a bunch of warehouses somewhere else. There's a postman who carries stuff between you and the warehouses. When you ask to load whatever address 3456, the postman goes to "3456 Memory Boulevard" and gets four packages (bytes) for you.

MM10 works by having the postman know that some addresses are special. If your LCD screen was mapped in memory at address 7890, and you try to send packages (bytes) to this address, the postman knows to deliver them to "7890 Video Road", not "7890 Memory Boulevard" (7890 Mem. Blvd. may not even exist!).

This topic also gets covered in lab 2, so the handout may be a good resource as well.
Recursive Calls

This may be one of those cases where a (moving) picture may be worth a thousand words, so there's an animation that lets you step through the Fibonacci program, which is the example we're using for a program with a recursive function. There's also a video walking through it; both should be available from the recitation notes page.

See you next Wednesday!
**6.004 Recitation IV**

**start:**
```
li a0, 4  # starting n is 4
```

**fibonacci:**  # Fibonacci(n)
```
li a1, 2
bge a0, a1, recurse
```

**base:**
```
li a0, 1
ret
```

**recurse:**
```
addi a0, a0, -1
addi sp, sp, -8  #
sw ra, 4(sp)     # ) push ra and a0
sw a0, 0(sp)     #
call fibonacci    # a0 ← Fibonacci(n-1)
mv a1, a0         # copy result to a1
lw a0, 0(sp)      #
lw ra, 4(sp)      # ) pop a0 and ra
addi sp, sp, +8   #
addi a0, a0, -1
addi sp, sp, -8   #
sw ra, 4(sp)      # ) push ra and a1
sw a1, 0(sp)      #
call fibonacci    # a0 ← Fibonacci(n-2)
lw a1, 0(sp)      #
lw ra, 4(sp)      # ) pop a1 and ra
addi sp, sp, +8   #
add a0, a0, a1
ret
```