Last of "how computers work"

Lecture 18

Machine code and more

Admin

- Exam scores
  - I still don't have scores from all of the GSIs.
  - It looks like the median and average are around 70, but I really can't tell yet.
  - I expect to have them all today, so I should be able to post the numbers and graphs later today.
  - Answers will be posted over the weekend.

Until break

- Our plans are as follows:
  - Monday and Wednesday will be an introduction to Matlab.
    - Matlab reading will be posted on the website.
  - Friday will cover some additional C++ language constructs and perhaps finish up Matlab (as needed)
  - The Monday we get back the next project will be discussed.

Last of "How a Computer works"

- We've covered some basics, but I felt a lot of you were lost.
  - So today I'm going to give a very detailed example of the execution of a computer.
  - The goal is to get you all to have some appreciation for what a computer is doing
  - The example is *very* simplistic, but at the same time does provide an accurate view of what a computer does.

Disclaimer

- This is an example of a *very* simplified computer. A real computer might have 100s of different instructions and around a billion bytes of memory. This one has 4 instructions and 64 bytes of memory.
  - Still, while simplified it does illustrate the basics of computer operation.

- Each computer instruction takes up 16 bits (2 bytes) of memory. The instructions are encoded as follows:

Instruction	**opcode [15:14]**	**memA[13:8]**	**B[7:0]**
add	00	memA	memB
addi	01	memA	immediate
beq	10	memA	target
print	11	memA	unused

Instruction	**opcode [15:14]**	**memA[13:8]**	**B[7:0]**
add	00	memA	memB
addi	01	memA	immediate
beq	10	memA	target
print	11	memA	unused

add: Mem[memA]=Mem[memA]+Mem[memB]
addi: Mem[memA]=Mem[MemA]+immediate
beq: if(Mem[memA]==0) PC=target
print: print Mem[memA] and halt.

So 0000 0000 0000 0000 says to add the byte at memory location 0 to itself and store the result in memory location zero.

Memory

- Memory is just a big array
  - Well not-so-big in our case.
- Each address contains a single byte (8 bits)
- The "memory address" is an index into the array.
  - Note that with 64 bytes of memory we need $\log_2(64)$ or 6 bits to address the memory.
  - 6 bits can represent 0 to 63.

PC

- The program counter, or "PC" points to the next instruction to be fetched.
  - It is just a special 6-bit memory location, separate from "memory"
- Recall that an instruction takes up 16 bit, so an instruction will use 2 memory addresses, PC and PC+1.
- When an instruction finished executing, it sets the PC = PC+2.
  - Branches will sometimes be an exception to this!

OK, let's look at an example:

- The algorithm

  sum=sum+x

  y=y-1

  if(y=0) done

```
start:  add sum, x
        addi y, –1
        beq y, done
        beq z, start
done:   print sum
sum:    0
y: 3
x: 4
z: 0
```

add: Mem[memA]=Mem[memA]+Mem[memB]
addi: Mem[memA]=Mem[MemA]+immediate
beq: if(Mem[memA]==0) PC=target
print: print Mem[memA] and halt.

Instruction	opcode [15:14]	memA[13:8]	B[7:0]
add	00	memA	memB
addi	01	memA	immediate
beq	10	memA	target
print	11	memA	unused

PC=

Address	Data
0	00001010
1	00001100
2	01001011
3	11111111
4	10001011
5	00001000
6	10001101
7	00000000
8	11001010
9	00000000
10	00000000
11	00000011
12	00000100
13	00000000

add: Mem[memA]=Mem[memA]+Mem[memB]
addi: Mem[memA]=Mem[MemA]+immediate
beq: if(Mem[memA]==0) PC=target
print: print Mem[memA] and halt.