First, a reminder....

- Computers communicate in bits
- Bit = electrical pulse
  - 0 = “off” or “low voltage”
  - 1 = “on” or “high voltage”

Instruction set

- Computer's *vocabulary*
- Considerations:
  - complexity (affects hardware)
  - size
  - regularity

Question of the day

What instructions should a computer contain?
Instruction set components

- Arithmetic
- Storage/data/variables/constants
- Logical
- Decision
- Procedures/functions/methods

Arithmetic

- Add
- Subtract
- The rest can be fashioned from these

Data storage

- Where?
- How much to allocate for storage?
- What if we run “over” the amount given?
- How do we access data?

Registers

- Memory locations to store data that our program will be using
  - operands of addition and subtraction
  - program counter
  - current instruction
- Typically, a small number, each of limited size
- Part of the hardware
Registers: MIPS

- Each register is 32 bits long
  - Q: what about large numbers? --> later
- 32 total registers
  - Fits into addressing scheme
  - More registers = longer distance for signals to travel = slower
- Registers named $sx$ or $tx$
  - $sx$ = program variables (numbered 16-23)
  - $tx$ = temporary storage (numbered 8-15)

Memory operands

- Load data from memory
- Store data into memory
- Array accessing
  - Note: word = 4 bytes
  - Therefore, addresses of sequential array elements differ by 4
  - Example: $A[0]$ is at address 0, $A[2]$ is at address 8, $A[8]$ is at address 32, ...

Memory: MIPS

- Array access: $offset(register)$
  - $offset$: position in array * 4
  - $register$: the register in which the “base address” of the array is stored (the address of the first element in the array)
- $lw$ $tx$, $offset(register)$: Loads the data at the specified array location into a temporary register
- $sw$ $tx$, $offset(register)$: Stores the data in the specified temporary register to the specified array location

Memory: MIPS examples

```
# loads $A[3]$ into memory, subtracts it from the # contents of $s0$, and stores the result back # into $A[3]$
lw $t0$, 12($s4) # assumes $A[0]$ address stored in $s4
sub $t0$, $s0$, $t0$
sw $t0$, 12($s4)
```