Big Idea: Bits can represent anything!

- Characters?
  - 26 letters => 5 bits ($2^5 = 32$)
  - Upper/lower case + punctuation => 7 bits (in 8) ("ASCII")
  - Standard code to cover all the world's languages => 8, 16, 32 bits ("Unicode")
    - www.unicode.com
  - Logical values?
    - 0 => false, 1 => true
  - Colors? Ex. Red(00) Green (01), Blue (11)
  - Locations, addresses, commands?
  - Memorize: N bits <=> at most 2^N things

How many bits to represent π ?

A) 1

B) 9 ($\pi = 3.14, so\ that's\ 011\ \text{“.”}\ 001\ 100$)

C) 64 (Since Macs are 64-bit machines)

D) Every bit the machine has!

E) ∞

RISC Pipelines

1) Fetch instructions from memory
2) Read registers and decode the instruction
3) Execute the instruction or calculate an address
4) Access an operand in data memory
5) Write the result into a register

RISC Instruction

“Ideally, each of the stages in a RISC processor pipeline should take 1 clock cycle so that the processor finishes an instruction each clock cycle and averages one cycle per instruction (CPI).”

“In practice, however, RISC processors operate at more than one cycle per instruction. The processor might occasionally stall a a result of data dependencies and branch instructions.”

Assembly Variables: Registers

- By convention, each register also has a name to make it easier to code
- 32 (32-bit) Registers
  - $0 => $31
- $16 - $23 => $s0 - $s7
  - (correspond to C variables)
- $8 - $15 => $t0 - $t7
  - (correspond to temporary variables)

Addition and subtraction of Integers

- How to do the following C Statement?
  - a = b + c + d – e;

- Break into multiple instructions
  - add $t0, $s1, $s2 #temp = b + c
  - add $t0, $t0, $s3 #temp = temp + d
  - sub $s0, $t0, $s4 # a = temp - e