The Stored Program Concept

- All programs (instructions) are just data represented by combinations of bytes!
- Any block of memory can be code. Consequently, self-modifying code is possible!
- The Program Counter (PC) is a special register (not directly accessible) which holds a pointer to the current instruction.

Instruction Formats

MIPS instructions come in three tasty flavors!

R-Instruction format (register-to-register). *Examples: addu, and, sll, jr*

op code	rs	rt	rd	shamt	funct
6 bits	5 bits	5 bits	5 bits	5 bits	6 bits

See green sheet to see what registers are read from and what is written to

I-Instruction Format (register immediate) Examples: addiu, andi, bne

op code	rs	rt	immediate
6 bits	5 bits	5 bits	16 bits

Note: Immediate is 0 or sign-extended depending on instruction (see green sheet)

J-Instruction Format (jump format) For j and jal

op code	address
6 bits	26 bits

KEY: An instruction is R-Format if the op code is 0. If the opcode is 2 or 3, it is J-format. Otherwise, it is I-format. Different R-format instructions are determined by the "funct".

1. How many instructions are representable with this format?

64 R-type instructions
63 other instructions
127 total

2. What could we do to increase the number of possible instructions?

Many things - for example, for R-type instructions where the shamt isn't used, use that to describe more instructions. Could delete one I-type instruction to make a second set of R-type instructions, etc. One trade off is the hardware needs to be more complex.

MIPS Addressing Modes

- We have several **addressing modes** to access memory (immediate not listed):
  - **Base displacement addressing**: Adds an immediate to a register value to create a memory address (used for lw, lb, sw, sb)
  - **PC-relative addressing**: Uses the PC (actually the current PC plus four) and adds the I-value of the instruction (multiplied by 4) to create an address (used by I-format branching instructions like beq, bne)
  - **Pseudodirect addressing**: Uses the upper four bits of the PC and concatenates a 26-bit value from the instruction (with implicit 00 lowest bits) to make a 32-bit address (used by J-format instructions)
  - **Register Addressing:** Uses the value in a register as memory (jr)

3. You need to jump to an instruction that is 257Mb up from the current PC. How do you do it? (HINT: you need multiple instructions)

```
addi    $t0, $pc, 0x10100000
jr      $t0
```

4. Given the following MIPS code (and instruction addresses), fill in the highlighted instructions (you'll need your green sheet!):

```
0x002cff00: loop: addu $t0, $t0, $t0      |  0 |  8 |  8 |  8 |  0 |  21  |

0x002cff04:       jal  foo                |  3 |          0xC0001         |

0x002cff08:       bne  $t0, $zero, loop   |  5 |  8  |  0  |     -0x3     |
...

0x00300004: foo:  jr $ra            $ra= 0x002cff08
```

5. What instruction is `0x00008A03`?

```
0x00008A03 = 000000 00000 00000 10001 01000 000011
Op: 000000 -> R-type          Rs: 0 (Unused)
Funct: 000011 -> sra          Rt: 0
Shamt: 01000 -> 8             Rd: 17
Final instruction: sra $17, $0, 8
```

Bonus Question: You have four numbers: a, b, c, and d. You know five of the six pair-wise products (ab, ac, ad, bc, bd, and cd) are 2, 3, 4, 5, and 6, but you don't know which five. What is the last pair-wise product?

12/5