Slide 1

Administrivia

- Reminder: Homework 2 due today. Homework 3 (by request) due Wednesday.

  (Some problems in Homework 2 seem ill-posed. ”Corrections” added to write-up, but it’s probably simplest to just answer them as asked, as if they made sense.)

  (Questions?)
- Quiz 3 next Monday.
- Appendix B has some additional information about MIPS assembler language. Section B.10 in particular has short descriptions of all instructions and also a table (p. 50) that maps opcode to instruction name.

Slide 2

Representing Data, Revisited

- To the hardware “it’s all ones and zeros”. But those ones and zeros can encode numbers (various forms), text, etc.
- Numbers in particular are interesting because we want to implement arithmetic operations.
- In theory you learned about integer representation and arithmetic in CSCI 1320. Review . . .

Slide 3

Binary Versus Decimal (Review?)

- In decimal (base 10) notation, each digit is multiplied by a power of 10. Same idea for binary (base 2), but using powers of 2.
- So, converting from binary to decimal is easy (if tedious), working from definition. Example?
- Converting from decimal to binary? Repeatedly divide by 2 and record remainders . . .

  We could describe this as a recursive algorithm for computing $bits(n)$:
  - Base case is $n < 2$; trivial.
  - For recursive step, divide n by 2 to get quotient q and remainder r. Then $n = 2q + r$, and:

    The last bit of $bits(n)$ should be r.

    The remaining bits are $bits(q)$, left-shifted by 1.
- Terminology: "Least significant" and "most significant" bits.

Slide 4

Binary Versus Hexadecimal (Review?)

- Binary is useful for showing real internal state but not very compact. Decimal is compact but not so easy to convert to/from binary.
- We might notice — easy to convert to/from a base that's a power of 2. Hence the use of "octal" (base 8) and "hexadecimal" (base 16). For the latter, we need more than 10 digits, so we use "A" through "F".

  Examples?
- Notice that we can also convert directly to/from decimal, much as we did for binary.

Slide 5

Representing Integers (Review?)

- Representing non-negative integers is easy — convert to binary and pad on the left with zeros.
- What about negative integers?
- Could try using one bit for sign, but then you have +0 and -0, and there are other complications.
- Or . . . consider a car odometer — in effect, representable numbers form a circle, since adding 1 to largest number yields 0.

Slide 6

Representing Integers, Continued (Review?)

- We could implement the car-odometer idea in binary, and then choose where to “cut the circle” (between smallest and largest):
  - Between 0 and all ones — unsigned integers.
  - Between largest number with 0 as the MSB and smallest number with 1 as MSB — “two’s complement” signed integers.
- Notice that with the two’s complement scheme, +1/-1 moves us “around the circle” — nothing special needed for negative numbers.
- Notice that if we have n bits, adding 2^n to x gives us x again. This leads to an easy way to compute $-x$: Compute $2^n - x$, and notice that

  $$2^n - x \;=\; (2^n - 1) - x + 1$$

  which is very easy to compute . . .

  Examples?

Slide 7

Minute Essay

- What are you finding interesting about the problems for chapter 2? what are you finding difficult?