MIPS Multiplication and division
Floating-point representation

Multiplication

- Series of adds and shifts
- Operands of \( n \) bits and \( m \) bits yield a product of \( n+m \) bits
- Shortcut: multiply by \( 2^n \) --> shift left by \( n \) bits
- MIPS implements by using 32 adders, parallelizing the additions
  - in each step, least significant bit goes to product
  - upper 31 bits plus carry becomes new operand

32-bit multiplication: algorithm

- If least significant bit of multiplier is 1, product = product + multiplicand
- Shift multiplicand left 1 bit
- Shift multiplier right 1 bit
- Repeat 32 times
- Variation: Place multiplier in least 32 bits of product register, product in upper half of register, shift right at each stage

Multiplication: example

- 1110 * 0010
- Multiplicand = 00001110, multiplier = 0010
- 1st pass:
  - mult. bit = 0, so product = 00000000
  - multc. = 00011100, mult. = 0001
- 2nd pass:
  - mult. bit = 1, so product = 00011100
  - multc = 00110000, mult. = 0000
- 3rd and 4th passes: mult. bit = 0, so no change in product
Multiplication

- Above algorithm will work for signed numbers too (just take lower 32 bits)
- MIPS:
  - 2 registers: Hi and Lo
  - mult, multu
  - mflo, mfhi: move result from Lo/Hi to general registers
  - overflows ignored

32-bit division: algorithm

- Divisor in upper 32 bits of divisor register, dividend in remainder register
- Remainder = Remainder – divisor
- If remainder > 0, shift quotient left and set rightmost bit of quotient to 1. Otherwise, add the divisor back to the remainder, shift quotient left and set rightmost bit of quotient to 0.
- Shift divisor right
- Repeat 33 times
- Variation: Use same register for dividend and remainder

Division: example

- 1010 / 0010
- Remainder = 00001010, divisor = 00100000
- 1st pass:
  - remainder – divisor < 0 (remainder stays the same)
  - quotient = 0000, divisor = 00010000
- 2nd pass:
  - remainder – divisor < 0 (remainder stays the same)
  - quotient = 0000, divisor = 00001000
- (cont.)

Division: example (cont.)

- 3rd pass:
  - remainder – divisor = 00000010
  - quotient = 0001, divisor = 00000100
- 4th pass:
  - remainder – divisor < 0 (remainder stays the same)
  - quotient = 0010, divisor = 00000010
- 5th pass:
  - remainder – divisor = 0
  - quotient = 0101, divisor = 00000001
Division notes

- Can't parallelize division, but can use lookup tables
  - "guess" the value to subtract
- MIPS:
  - `div`, `divu`
  - Lo/Hi registers
  - does not detect overflow or divide by 0

Floating point numbers

- Scientific notation
- Form: $1.xxx... \times 2^{yyy}$
  - $xxx$: mantissa (precision)
  - $yyy$: exponent (size)
  - `normalized` format
- Sign magnitude format
- Note: leading 1 is assumed
  - 000....000 reserved for 0

IEEE 754 floating-point standard

- Standard for storing floating-point numbers
- Single-precision:
  - 1 sign bit
  - 8 exponent bits
  - 23 mantissa bits
- Double-precision:
  - 1 sign bit
  - 11 exponent bits
  - 52 mantissa bits