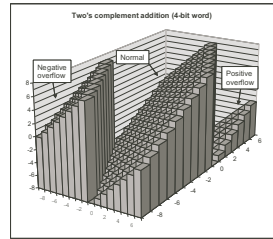






## Overflow

- overflow occurs when the result cannot be represented in the given number of bits, because the magnitude is too great [O'H&Bryant, fig 2.14]: 4-bit 2's c. add



- rules:
  - $+x + -y \rightarrow$  no overflow
  - $-x + +y \rightarrow$  no overflow
  - $+x + +y \rightarrow$  overflow if result is  $-z$
  - $-x + -y \rightarrow$  overflow if result is  $+z$
- example:

$$\begin{array}{r} 0\ 0001 + \quad 1 + \\ 0\ 1111 = \quad 15 = \\ 1\ 0000 \quad 0 \end{array}$$

- discussion point: what are the possible consequences? when should it be checked?

## Review Exercise: Number Conversion

- binary  $\rightarrow$  hexadecimal  $\rightarrow$  decimal:

$$\begin{aligned} 0001\ 0111\ 1010\ 0101_2 &= 17A5_{16} \\ &= 1 \times 16^3 + 7 \times 16^2 + 10 \times 16^1 + 5 \\ &= 4096 + 1792 + 160 + 5 \\ &= 6053 \end{aligned}$$

- for large numbers, the above may be easier to do (by hand) than direct conversion:

$$\begin{aligned} 0001\ 0111\ 1010\ 0101_2 &= 1 \times 2^{12} + 1 \times 2^{10} + 1 \times 2^9 + 1 \times 2^8 + 1 \times 2^7 + 1 \times 2^5 + 1 \times 2^2 + 1 \times 2^0 \\ &= 4096 + 1024 + 512 + 256 + 128 + 32 + 4 + 1 \\ &= 6053 \end{aligned}$$

note: we have skipped the 0's (e.g.  $0 \times 2^{15}$ ) above

- in general, hexadecimal is the most widely used system in computer systems (after decimal)

## Sign Extension

- example: we want to convert a 10-bit number into a 16-bit number (needed later for PeANUt operands)

- easy for positive case:

$$\begin{array}{r} 0110\ 6 \\ 0000\ 0110\ 6 \end{array}$$

- negative case?

$$\begin{array}{r} 1010\ -6 \\ 0000\ 1010\ 10 \end{array}$$

- solution: fill the extra digits with copies of the sign bit (the leftmost bit)

- positive case:

$$\begin{array}{r} 0111\ 7 \\ 0000\ 0111\ 7 \end{array}$$

- negative case:

$$\begin{array}{r} 1001\ -7 \\ 1111\ 1001\ -7 \end{array}$$