```
Recitation Outline 9/08/03 213


- C Commenting Style
  - Dont do this!
    int bitAnd(int x, int y){
      /*
        ~x and or it with ~y and then ~ the whole thing in order to get
        x & y.
      */
      return ~(~x|~y);
    }

    Do This:
    int bitAnd(int x, int y){
      /* De Morgan's Law */
      return ~(~x|~y);
    }

- Simple Bit Conversions

  - 2's Compliment representation
    - Positive numbers: same as unsigned numbers
    - Negative numbers: negate the positive part and add 1
            -x = ~x + 1

    - Range: For n-bit space: -2^(n-1) to (2^(n-1))-1
      e.g. 8-bit: -128 to 127 (TMin to TMax)

  - Fill in the table:
  ----------------+----------+---------------+----------+
  Special Meaning |  Decimal |   2's Comp    |    Hex   |
  ----------------+----------+---------------+----------+
                  |     0    |    00000000   |   0x00   |
                  |     1    |   s00000001   |  s0x01   |
                  |    45    |   s00101101   |  s0x2D   |
                  |   s101   |    01100101   |  s0x65   |
                  |   s30    |   s00011110   |   0x1E   |
      TMax        |   s127   |   s01111111   |  s0x7F   |
                  |   s-119  |    10001001   |  s0x89   |
                  |   s-124  |   s10000100   |   0x84   |
                  |    -1    |   s11111111   |  s0xff   |
                  |   -95    |   s10100001   |  s0xA1   |
      TMin        |  s-128   |   s10000000   |  s0x80   |
  ----------------+----------+---------------+----------+

- Biasing
  - Lecture 9/02 slides 34-37

- Bit Puzzles
  - Absolute Value
  int abs(int x){
    int sign = x >> 31;
    int neg_x = ~x+1;
    return (neg_x & sign) | (x & ~sign);
  }

  int abs(int x){
    int sign = x >> 31;
    return (x^sign)+(sign&1);
  }

  - vector add
  /* assume a + b + c + d <= 127
            a,b,c,d all positive
```

```
     return a + b + c + d using only 2 additions
  */
  int add4( int a, int b, int c, int d){
    x = a << 8 | b;
    y = c << 8 | d;
    int q = x + y;
    return (q & 0xff) + (q >> 8);
  }

  - Parity
      /* Problem: return 1 if x contains an odd number of 1s, 0 otherwise.
      * Operation bound: 20 ops (same ops as in Lab 1)
      *
      * Approach:
      *   Can use XOR to evaluate the parity of a 2-bit number.
      *   XOR: the XOR operation evaluates the parity
      *   2 bits: 0 XOR 1 is 1 (so returns 1 for 01 and 10)
      *   what we want is XOR of all 32 bits of x
      *   too many ops! (31)
      *   Solution: Do things in parallel
      *
      *   Draw picture of doing XOR by cutting and parallel bit-wise XOR
      */


     int bitParity(int x) {
       int wd16 = x ^ (x >> 16);
       int wd8 = wd16 ^ (wd16 >> 8);
       int wd4 = wd8 ^ (wd8 >> 4);
       int wd2 = wd4 ^ (wd4 >> 2);
       int bit = wd2 ^ (wd2 >> 1);

       return bit & 0x1;
     }
```