CRC Computations

Alex Breslow
Example of CRC-8

- polynomial is $x^8 + x^2 + x + 1$
- express as 100000111
- Let's say we want to know 1111100000 divided by our polynomial

\[
\begin{array}{c|c}
100000111 & 1111100000 \\
100000111 & \\
\hline
111101110 & \\
100000111 & \\
\hline
11101001 & \\
\end{array}
\]
Example of CRC-8

- polynomial is $x^8 + x^2 + x + 1$ (k=8)
- express as 100000111
- Let’s say we want to know 1111100000 divided by our polynomial

\[
\begin{array}{c|c}
100000111 & 1111100000 \\
100000111 & \text{XOR} \\
\hline
111101110 & \text{shift left by 1} \\
100000111 & \text{XOR} \\
\hline
11101001 & \text{stop since we are left with 8 bits}
\end{array}
\]

Pseudocode

Is the most significant bit in what remains 0?
If yes, shift what remains left until the leading bit is a 1.
If not, XOR the divisor with what remains.
Repeat until we have a remainder (number <= k bits long)
Example of CRC-8

// Function returns the remainder from a CRC calculation on a char* array of length byte_len
char crc8(char* array, size_t byte_len){
    const char poly = shift_right_by_one(x^8 + x^2 + x^1 + 1);
    char crc = array[0];
    int i, j;
    for(i = 1; i < byte_len; i++){  
        char next_byte = ith byte of array;
        for(j = 7; j >= 0; j--){ // Start at most significant bit of next byte and work our way down
            if(crc’s most significant bit is a 0){
                shift_left_by_one(crc); // Shift out 0
                crc = crc OR get_bit(j, next_byte); // Shift in next bit
            }
            else{ // crc’s most significant bit is a 1
                crc = shift_left_by_one((crc XOR (poly)); // Do first 8 bits of modulo 2 subtraction
                crc = crc OR (get_bit(j, next_byte) XOR 1); // Do final bit of modulo 2 subtraction
                } // and place in position 0 of crc register
        }
    }
    return crc;
}
You’ll need to implement the following:

- `char get_bit(int pos, char byte);` // Return a char with a value of 0 or 1 depending on whether the bit in the pos most significant bit is 0 or 1
  - So if byte has a value of 0x08 or 00001000, then for any value of pos between 0 and 7 other than 3, the return value should be 0 and otherwise 1.

- `shift_right_by_one` -- implement by using the right shift operator in C

- `shift_left_by_one` -- implement by using the left shift operator in C
Refresher on Bit Operators in C

- Assume integers X, Y, Z of an equal number of bytes, i.e. int X, Y, Z;

- X OR Y      ----> X | Y
- X AND Y     ----> X & Y
- X XOR Y     ----> X ^ Y

- NOT X       ----> ~X
- Shift X by Y bits to the left      ----> X << Y
- Shift X by Y bits to the right    ----> X >> Y

- Example: Z = Z XOR (X shifted to the left by Y bits)
  - Z ^= X << Y; // ^= operator is combo of XOR and assignment (=)

- Example: X = (X AND (NOT Y))
  - X = X & (~Y);

- See [http://www.tutorialspoint.com/cprogramming/c_bitwise_operators.htm](http://www.tutorialspoint.com/cprogramming/c_bitwise_operators.htm)