NUMBERS
Number Systems

- Generally:
  \[ X_b = a_{n-1} \times b^{n-1} + a_{n-2} \times b^{n-2} + \ldots + a_0 \times b^0 \]

- Decimal:
  \[ 2345_{10} = 2 \times 10^3 + 3 \times 10^2 + 4 \times 10^1 + 5 \times 10^0 \]

- Binary:
  \[ 1011_2 = 1 \times 2^3 + 0 \times 2^2 + 1 \times 2^1 + 1 \times 2^0 \]

- Digital systems consist of elements which generally have only two possible states. Therefore the binary number system is a natural choice of number system.
Bits and Bytes

• In general, number systems start at 0 not 1

• 1 Bit means 1 Binary Digit (represented by a 1 or 0)

• 8 Bits is referred to as 1 Byte
  • (00000000 - 11111111 = 0 – 255, The range is $2^8 - 1$)

• 16 Bits is called 1 Word

• 4 Bits is 1 Nibble
Hexadecimal

• Hexadecimal is the number system using base 16
• Each digit can take 16 possible values ( 0 – 9, A - F )
• Each Hex digit is equivalent to 4 Bits in Binary

• \( 8A \text{ Hex} = 10001010 \text{ binary} = 138 \text{ decimal} \)

• Why? 4 binary digits can have \( 2^4 = 16 \) possible values
• Which is the same as a single Hex digit

• Therefore, Hex is a nice compact way of representing binary numbers (and is therefore often used when programming)
Signed Numbers

• In Binary number systems, if signed arithmetic is necessary, the most significant bit is used to represent the sign.

  • 00110010 is positive
  • 10110010 is negative

• For 1 Byte, Number Range is therefore – $2^7$ to $2^7-1 = -128$ to +127 (Not $2^8$)
Two’s Complement Arithmetic

- Consider a 3 bit code (a 3 bit computer?)
- +3: 011 binary
- Complement: 100
- Comp + 1: 101 (This is the Two’s complement for -3)
- Addition: (+3) 011
  + (-3) 101
  = 0 1000
  Lost!

- The 68HC11 has special instructions for creating 2’s complement numbers.
Real Numbers

• How do you represent fractional numbers using binary?

• Eg 1 0 1 1 . 1 1 0 1
  \[2^3 2^2 2^1 2^0 . 2^{-1} 2^{-2} 2^{-3} 2^{-4}\]
  = 11.8125

• In the above notation, the range is very small (around 2^4)

• Normally, scientific notation is used: p x b^q
  – P = Mantissa or Significand
  – B = Base
  – Q = Exponent
Real Numbers (2)

• Example: 1 Byte Mantissa, 1 Byte Exponent (base 10)

\[
\begin{align*}
01010010 & \quad 00000010 \\
0.640625 & \quad 2 \\
= 0.640625 \times 10^2 & \\
= 64.0625
\end{align*}
\]

• A much larger range is possible using this method

• Many representations are possible the 68HC11 uses its own format

• The most common format is defined in IEEE 754
Representing Characters - ASCII

- From: http://www.mindspring.com/~jc1/serial/Resources/ASCII.html

|     | 0   | 1   | 2   | 3   | 4   | 5   | 6   | 7   | 8   | 9   | A   | B   | C   | D   | E   | F   |
|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|
| 0   | NUL | SOH | STX | ETX | EOT | ENQ | ACK | BEL | BS  | HT  | LF  | VT  | FF  | CR  | SO  | SI  |
| 1   | DLE | DC1 | DC2 | DC3 | DC4 | NAK | SYN | ETB | CAN | EM  | SUB | ESC | FS  | GS  | RS  | US  |
| 2   | SPC | !   | "   | #   | $   | %   | &   | '   | (   | )   | *   | +   | ,   | -   | .   | /   |
| 3   | 0   | 1   | 2   | 3   | 4   | 5   | 6   | 7   | 8   | 9   | :   | ;   | <   | =   | >   | ?   |
| 4   | @   | A   | B   | C   | D   | E   | F   | G   | H   | I   | J   | K   | L   | M   | N   | O   |
| 5   | P   | Q   | R   | S   | T   | U   | V   | W   | X   | Y   | Z   | [   | \   | ]   | ^   | _   |
| 6   | `   | a   | b   | c   | d   | e   | f   | g   | h   | i   | j   | k   | l   | m   | n   | o   |
| 7   | p   | q   | r   | s   | t   | u   | v   | w   | x   | y   | z   | {   | |   | }   | ~   | DEL |
Logical Operations

- **AND**
  
  \[
  \begin{array}{c}
  01100111 \\
  10100101 \\
  \hline
  00100101 \\
  \end{array}
  \]
  
  = 00100101

- **OR**
  
  \[
  \begin{array}{c}
  01100111 \\
  10100101 \\
  \hline
  11100111 \\
  \end{array}
  \]
  
  = 11100111

- **XOR**
  
  \[
  \begin{array}{c}
  01100111 \\
  10100101 \\
  \hline
  11000010 \\
  \end{array}
  \]
  
  = 11000010

- These are the only three logical operations supported by the HC11
Bit Masks

• If I want to know the value of the low nibble of a byte:

\[
\begin{align*}
11010011 & \quad \text{AND} \\
00001111 & \quad \text{=} \\
00000111 & \quad \text{=} 
\end{align*}
\]

• If I want to set one bit:

\[
\begin{align*}
11010011 & \quad \text{OR} \\
00000100 & \quad \text{=} \\
11010111 & \quad \text{=} 
\end{align*}
\]

Friday, 27 January 12
Numbers