Aufgabe 1: Lernen Sie das Byte kennen

Schauen Sie sich zunächst die Byte-Darstellung genau an. Das Byte wird in Byte-Fiddler von links nach rechts gelesen. Dies wird an dieser Stelle erwähnt, weil es auch die umgekehrte Darstellungsform gibt.
Spielen Sie ein wenig mit dem Byte herum und probieren sie alle Funktionen aus. Auch wenn sich einige Zusammenhänge nicht sofort erschließen.

Achten Sie nun auf die Dezimaldarstellung. Vielleicht fällt Ihnen auf, dass die Bits unterschiedliche Wertigkeiten haben. Das mag zunächst ungewohnt erscheinen, aber Sie kennen diesen Zusammenhang bereits aus dem Ihnen vertrauten Dezimalsystem.

Will man im Dezimalsystem eine Zahl darstellen, die größer ist als 9, macht man das, indem man die nächst höhere Stelle um ein erhöht und die 9 auf 0 “zurücksetzt”.

$9_{10}+1_{10} = 10_{10}$

Genau so funktioniert es auch im Dualsystem. Nur gibt es statt den zehn Ziffern (0-9) nur die Ziffern 0 und 1. Will man die Zahl zwei darstellen, muss man bereits auf einen Übertrag zurückgreifen.

$1_2+1_2 = 10_2$

Möchte man den Wert einer Zahl ermitteln, kann man in Stellenwertsystemen den Wert der Stelle mal die Anzahl der Ziffern hoch die Stelle biginnend mit 0 rechnen.

Für die Dualzahl 1001 0111 geht das so:

$1*2^7 + 0*2^6 + 0*2^5 + 1*2^4 + 0*2^3 + 1*2^2 + 1*2^1 + 1*2^0 = 128 + 0 + 0 + 16 + 0 + 4 + 2 + 1 = 151_{10}$

Genau so verhält es sich auch im Zehner-System:

$1*10^2 + 5*10^1 + 1*1^0 = 100 + 50 + 1 = 151_{10}$

a) Bilden Sie im ByteFiddler folgende Zahlen: 128, 10, 11, 151, 217, 255, 123, 137, 136
b) Sie kennen nun bereits einen Weg, Dualzahlen in das Dezimalsystem zu überführen. Recherchieren Sie im Internet, wie man Dezimalzahlen in das Dualsystem überführt.
c) Wieviele unterschiedlichen Werte kann ein Byte maximal annehmen?
d) Wie kann man binär negative Zahlen darstellen ohne ein Minuszeichen?

Aufgabe 2: Die Zahl 16 - Hex und Nibble

Mit dem Dualsystem sind sie nun vertraut. Bits und Bytes sind für Sie nun nichts neues mehr. In der Informatik macht man sich an vielen Stellen auch das Hexadezimalsystem zur Nutze. Es hat statt 10 Ziffern, 16 Ziffern. Auf die 9 folgen hier die Buchstaben A,B,C,D,E und F. Das wirkt zunächst etwas seltsam, beim nähreren hinsehen ist das jedoch wirklich nützlich.
Um zu verstehen, warum "Hexadezimal" so hilfreich ist, sollten sie sich das Byte nochmal ansehen: Unterteilt man ein Byte in zwei Teile hat man zwei sogennte Nibble. ein Nibble besteht aus vier Bit. Ein Nibble hat 16! unterschiedliche Werte, die es annehmen kann. Speicherabbilder werden oft hexadezimal dargestellt, weil es sich gut lesen lässt.

Machen sie nun erste Erfahrungen damit:

A) Geben Sie folgende Bitmuster in ByteFiddler ein: 1111 1111, 0000 0000, 1001 1001, 0010 0010, 0101 0101, 0000 1111, 1111 0000
Achten Sie dabei auf die Hex-Darstellung. Was fällt ihnen auf?

Wahrscheinlich haben Sie nun eine Ahnung, warum das hexadezimale System nützlich sein kann. Die Berechnung aus Aufgabe 1 funktioniert natürlich auch im hexadezimalen System:

$FA23_{16} = 15 * 16^3 + 10 * 16^2 + 2*16^1 + 3*16^0 = 61440 + 2560 + 32 + 3 = 64035_{10}$

Aufgabe 3: Logische Operatoren

Wechseln Sie nun im ByteFiddler in den Calculate Modus. Recherchieren Sie zu logischen Operatoren. Interessant sind vor allem: “OR”, “XOR” und “AND”

Im folgenden werden Ihnen nun Bitmuster vorgegeben, die Sie in den ersten Operanden eingeben sollen um ihn mit einer logischen Operation und dem zweiten Operanden zum gewünschten Ergebnis manipulieren sollen.

Beispiel: Sie geben 1010 1000 ein und stellen den Operator auf “OR”. Um als Ergebnis nun 1010 1111 zu erhalten, müssten Sie im zweiten Operanden zum Beispiel 0000 0111 eingeben (hier wären auch andere Lösungen denkbar).

A) Erzeugen Sie die Werte im ByteFiddler und tragen Sie ihre Erebnisse in die Tabelle ein.

Operand 1	Operator	Operand 2	Result
0001 0101	OR		1101 0111
0101 1010	OR		0101 1011
0000 0000	OR		1100 1100
0011 1100	XOR		1100 1100
1111 1111	XOR		1010 0000
0000 1010	XOR		1010 0000
0101 1111	AND		0000 0000
0101 1100	AND		0101 1100
1110 1101	AND		1100 1001

B) Mit welchem Operator können Sie Bits auf 1 stellen?
C) Mit welchem Operator können Sie Bits drehen/umkehren?
D) Mit welchem Operator können Sie Bits auf 0 stellen?
E) Zuletzt stellen Sie den DIV-Operator ein und setzen das höchste Bit. Schauen Sie was passiert, wenn von rechts ein Bit immer weiter an das höchste Bit nach links “geschoben” wird (man sagt auch shiften).