Teil 1: DATEN CODIEREN AUFGABEN

Aufgaben zu numerischen Codes

1. Ergänzen sie die Tabelle. Studieren sie anschliessend ihre fertig ausgefüllte Tabelle, insbesondere die Kolonnen mit den Binärwerten. Was stellen sie fest?

DEC	HEX	BIN 2^3	BIN 2^2	BIN 2^1	BIN 2^0
0	**0**	**0**	**0**	**0**	**0**
1	1	0	0	0	1
2	2	0	0	1	0
3	3	0	0	1	11
4	4	0	1	0	0
5	5	0	1	0	1
6	6	0	1	1	0
7	7	0	1	1	1
8	**8**	**1**	**0**	**0**	**0**
9	9	1	0	0	1
10	A	1	0	1	0
11	B	1	0	1	1
12	C	1	1	0	0
13	D	1	1	0	1
14	E	1	1	1	0
15	**F**	**1**	**1**	**1**	**1**

2. Wandeln sie die folgende Dezimalzahl ohne Taschenrechner in die entsprechende Binärzahl um: **911** $911_D = 0011'1000'1111_B$

3. Wandeln sie die folgende Binärzahl ohne Taschenrechner in die entsprechende Dezimalzahl um: **1011'0110** $1011'0110_D = 182_B$

4. Wandeln sie die folgende Binärzahl ohne Taschenrechner in die entsprechende Hexadezimalzahl um: **1110'0010'1010'0101** $1110'0010'1010'0101_B = E2A5_H$

5. Was ergibt die Addition der beiden binären Zahlen 1101'1001 und 0111'0101? Beachten sie, dass für das Resultat ebenfalls nur 8 Binärstellen zur Verfügung stehen.

1101'1001 Zahl1
0111'0101 Zahl2
1110'0010 Übertrag

0100'1110 Summe Dataoverflow!

6. Was könnten die beiden folgenden binären Wert für eine Bedeutung haben?
(*Tipp: a. ins Dezimalsystem umrechnen, b. ins Hexadezimalsystem umrechnen*)
   a. **1100'0000.1010'1000.0100'1100.1101'0011** 192.168.76.211 = Private IPv4-Adresse
   b. **1011'1110-1000'0011-1000'0101-1101'0101-1110'0100-1111'1110**
   BE-83-85-D5-E4-FE = MAC-Adresse in HEX

7. *Optional für Linux-Fans*: Was könnte die folgende in einem Bash-Script entdeckte Zeile für eine Bedeutung haben? Hinweis: Hier handelt es sich um das oktale Zahlensystem. **chmod 751 CreateWeeklyReport**
rwx rwx rwx owner group other Zugriffsrechte für File-owner:rwx, group:rx, other:x
007 005 001
111 101 001

8. Dimensionieren sie für den Matterhorn-Express, wo insgesamt 107 Gondeln die Touristen von Zermatt auf den Trockenen-Steg befördern, die Codebreite des Binärcodes für die Kabinenzählung. 107 Gondeln: log107/log2=6.74 Somit 7Bit (2^7=128)

9. *Optionale Aufgabe*: Sie untersuchen einen Arbeitsspeicher mit 12-Bit-Adress- bzw. 16-Bit-Datenbus.
   Welche Speicherkapazität in kiB besitzt dieser? (Hinweis: 1kiB=1024B)
   12Bit Adresse ergibt 2 12 Speicherplätze oder 4096. Pro Speicherplatz 2B ergibt 8192Byte. 8192/1024= 8kiBi
10. *Optionale Aufgabe*: Zwei Geräte sind mit einer seriellen Leitung und zusätzlichem Taktsignal verbunden. Das Taktsignal beträgt 1MHz.
    a. Wie viele Bytes können damit pro Sekunde übertragen werden?
       1MHz = 1'000'000 Hz. Somit 1'000'000Bit pro Sekunde oder 125kB/s
    b. Wie viele Bytes pro Sekunde könnten übertragen werden, wenn die Verbindung der beiden Geräte nicht seriell, sondern 8 Bit-parallel wäre?
       8x mehr. Somit 1MB/s
11. Bei den bisherigen Aufgaben zu Binärcodes, handelte es sich immer um positive, numerische Werte. Bei den Programmiersprachen spricht man vom Datentyp "unsigned integer". Hin und wieder möchte man aber auch die negative Zahlenwelt miteinbeziehen, man nennt dies dann "signed integer", was mit vorzeichenbehafteter Ganzzahl übersetzt werden kann.
    (*Wobei man gut beraten ist, abzuwägen, ob der Mehraufwand für negative Zahlen tatsächlich gerechtfertigt ist, wie folgendes Beispiel aus der Physik zeigt: Die Celsius-Temperaturskala kennt negative Werte, die Kelvin-Temperaturskala hingegen nicht und beide Skalen messen dieselbe Temperatur. Zum Beispiel entspricht der absolute Temperatur-Nullpunkt (nichts kann kälter sein, dass ist in diesem Fall entscheidend) -273 Grad Celsius aber eben auch 0 Grad Kelvin, Wasser schmilzt bei 0 Grad Celsius bzw. 273 Grad Kelvin und der heutige Sommer bot angenehme 35 Grad Celsius oder eben 308 Grad Kelvin*)
    Möchte man den binären Zahlenstrahl in den negativen Bereich erweitern, liegt die Versuchung nahe, das erste Bit (MSB) als Vorzeichen zu verwenden. Funktioniert leider nicht bzw. nicht in allen Fällen! Die Lösung die funktioniert, nennt man Zweierkomplement. Was darunter zu verstehen ist, wird im Internet unzählige Male erklärt.
    Nun zur Aufgabe: Wir gehen von einer Verarbeitungsbreite von einem Byte aus. (Datenbusbreite:1Byte)
    a. Nennen sie kleinster und grösster Binärwert bzw. Dezimaläquivalent im Falle von unsigned bzw. Vorzeichenlos.
       0000'0000=0 bis 1111'1111=255
    b. Nennen sie kleinster und grösster Binärwert bzw. Dezimaläquivalent im Falle von signed bzw. Vorzeichenbehaftet.
       1000'0000 bis 0111'1111
       -128 bis +127
    c. Wandeln sie die Dezimalzahl +83 in einen vorzeichenbehafteten Binärwert um. (signed)
       0101'0011 (=+83)
    d. Wandeln sie die Dezimalzahl -83 in einen vorzeichenbehafteten Binärwert um. Signed mit 2er-Komplement:
       0101'0011 +83
       1010'1100 um 1
       1010'1101 (=-83)
    e. Addieren sie die beiden erhaltenen Binärwerte zusammen. Es sollte 0 ergeben!

       0101'0011 +
       1010'1101
       1111'1110 Übertrag
       ----
    f. Wandeln sie die Dezimalzahl 0 in einen vorzeichenbehafteten Binärwert um. (signed). Hat ihre vorangegangene Addition auch diesen Binärwert ergeben?
       0000'0000 Ja!
    g. Warum können sie bei der gegebenen Datenbusbreite von 1 Byte die Dezimalzahl +150 nicht in einen vorzeichenbehafteten Binärwert umwandeln?
       (*Ziehen sie daraus ihre Lehren für zukünftige Programmiersprachkurse: Immer den korrekten Datentyp in der verlangten Grösse wählen*)
       Sprengt die Bitbreite von 8 Bit

12. Bisher haben wir immer von ganzen Zahlen gesprochen. Oft genügt das in der realen Welt aber nicht. Dazu ein Beispiel: Teile ich die Ganzzahl 1 durch die Ganzzahl 3 und multipliziere sie darauf wieder mit der Ganzzahl 3 erhalte ich, sofern der Compiler/Interpreter nicht trickst, die Ganzzahl 0, was bekanntlich falsch ist. Dies weil das Resultat der Division nicht als 0.3333333 sondern als ganze Zahl 0 (Nachkommastellen werden ignoriert) zwischengespeichert wird. Benötigt wird also ein Datentyp, der mit Fliesskommazahlen (Floating Point Numbers) klarkommt. Wie würden sie eine solche Fliesskommazahl definieren, und wie sie digital abspeichern? Machen sie dazu einen Vorschlag.

13. *Optionale Aufgabe*: Erstellen sie die Wahrheitstabellen für die folgenden Funktionen:
    a. Logisch UND/AND (*mit zwei Eingangs- und einer Ausgangsvariablen*)

A	B	Y AND
0	0	0
0	1	0
1	0	0
1	1	1

    b. Logisch ODER/OR (*mit zwei Eingangs- und einer Ausgangsvariablen*)

A	B	Y OR
0	0	0
0	1	1
1	0	1
1	1	1

    c. Logisch NICHT/NOT (*mit einer Eingangs- und einer Ausgangsvariablen*)

A	Y NOT
0	1
1	0

    d. Logisch EXOR (*mit zwei Eingangs- und einer Ausgangsvariablen*)

A	B	Y EXOR
0	0	0
0	1	1
1	0	1
1	1	0

14. *Optionale Aufgabe*: Eine in der Computertechnik wichtige mathematische Funktion ist die Restwert- oder Modulo-Funktion mit dem in z.B. Java und C verwendeten Operationszeichen %. Versuchen sie nun die folgende Berechnungen auszuführen. Was stellen sie fest?
    a. 11 % 2 = 1
    b. 10 % 2 = 0
    c. 10 % 3 = 1
    d. 10 % 5 = 0
    e. 10 % 9 = 1

Aufgabe zu Notepad++ und HxD

Sie erhalten eine ZIP-Datei switzerland.zip unter folgendem Link:
https://juergarnold.ch/Codesysteme/switzerland.zip

Laden sie die ZIP-Datei auf ihren Notebook und extrahieren sie die binäre Datei switzerland.bin. Untersuchen sie diese mit Notepad++ und HxD. In der Datei verstecken sich vier vorzeichenlose 16-Bit-Ganzzahlen. Jede davon hat einen Bezug zur Schweiz. Wandeln sie die vier Zahlen je in ihr Dezimaläquivalent um und geben sie einen Tipp ab, was deren CH-Bedeutung sein kann.

Aufgaben zu alphanumerischen Codes

Suchen sie sich für die folgenden Aufgaben im Internet nach einer Webseite, wo Unicode-Zeichen gelistet sind.

1. Sie erhalten eine ZIP-Datei Textsamples.zip unter folgendem Link:
   https://juergarnold.ch/Codesysteme/Textsamples.zip
   Laden sie die ZIP-Datei auf ihren Notebook und extrahieren sie die drei Dateien **`Textsample1`**, **`Textsample2`** und **`Textsample3`**. Eine der drei Dateien ist in ASCII codiert, die andere in UTF-8 und die dritte in UTF-16. Beantworten sie nun die folgenden Fragen:

   a. Welche der Dateien ist nun ASCII-codiert, welche UTF-8 und welche UTF-16 BE-BOM?

   b. Alle drei Dateien enthalten denselben Text. Aus wie vielen Zeichen besteht dieser?

   c. Was sind die jeweiligen Dateigrössen? (Beachten sie, dass unter Grösse auf Datenträger jeweils 0 Bytes angegeben wird. Dies darum, weil beim Windows-Dateisystem NTFS kleine Dateien direkt in die MFT (Master File Table) geschrieben werden.) Wie erklären sie sich die Unterschiede?

   d. Bei den weiteren Fragen interessieren uns nur noch die ASCII- und die UTF-8-Datei: Bekanntlich ist UTF-8 in den ersten 128 Zeichen deckungsgleich mit ASCII. Untersuchen sie nun die beiden HEX-Dumps und geben sie an, welche Zeichen unterschiedlich codiert sind. Ein kleiner Tipp: Es sind deren zwei.

   e. Was bedeuten die beiden Ausdrücke, denen wir z.B. bei UTF-16 begegnen: Big-Endian (BE), Little-Endian (LE)?

   f. Im Notepad++ kann man unter dem Menüpunkt Codierung von ASCII zu UTF umschalten. Spielen sie damit etwas herum und notieren sie sich, was in der Darstellung jeweils ändert.

   g. Für Anspruchsvolle: Der UTF-8-Code kann je nach Zeichen ein, zwei, drei oder vier Byte lang sein. Wie kann der Textreader erkennen, wann ein UTF-8 Zeichencode beginnt und wann er endet? Untersuchen sie dies anhand der beiden Textsamples und lesen sie in z.B. Wikipedia die entsprechende Theorie zu UTF-8 durch. Tipp: Startbyte und Folgebyte.

2. *Optionale Aufgabe*: **Einfacher QR-Code erstellen und lesen**:
   Denken sie sich eine kurze Botschaft, URL etc. aus und bilden sie diese Information in einem QR-Code ab. Danach tauschen sie mit ihrem Banknachbar/in ihre QR-Codes aus. Wenn der QR-Code gelesen werden kann, waren sie erfolgreich. Applikationen die alphanumerischen Text in einen QR-Code und zurück wandeln, findet man im Internet.

3. *Optionale Aufgabe*: **Zusammengesetzte Daten in einem QR-Code**:
   Ihre Firma wird mit dem Design von Tickets für ein Fussballstadion beauftragt. Das Eintrittsticket soll mit einem QR-Code versehen sein, der alle wichtigen Informationen zur Buchung enthält. Dies soll ermöglichen, das Ticket jederzeit und überall von Offline-QR-Readern lesen und überprüfen zu lassen bzw. Zugang zu den Stadionbereichen zu gewähren. Das Ticket soll die folgenden, codierten Informationen enthalten:

   a. Datum und Uhrzeit der Veranstaltung
   b. Fortlaufende, alphanumerische Ticketnummer (Ticketnummern seit Tag-0)
   c. Numerische Sitzplatznummer (Jeder Sitzplatz hat eigene Nummer)
   d. Tribünensektor A-Z (Für einfache Platzeinweisung der Besucher)
   e. ID- oder Passnummer des Besuchers (Tickets nicht übertragbar)

   - Überlegen sie sich, ob das Ticket fälschungssicher ist. Braucht es allenfalls noch eine Prüfziffer, Checksum etc.?
   - Wenn der Besucher am Stadioneingang erscheint, zeigt er sein Ticket. Dieses wird von einem Platzanweiser mit einem QR-Code-Leser gelesen. Damit kann er die Personalie und Gültigkeit des Tickets überprüfen und den Besucher anschliessend in den richtigen Stadionsektor leiten.
   - Bilden sie diesen Datensatz in einen QR-Code ab. Erstellen sie QR-Codes für fiktive Veranstaltungen. Testen sie ihre QR-Codes mit ihrem Banknachbarn/in gegenseitig aus.
   - Applikationen die alphanumerischen Text in einen QR-Code und zurück wandeln, findet man im Internet.
   - Auf Komplettlösungen aus dem Internet - es gibt zu diesem Thema mehr oder weniger "pfannenfertige" Applikationen - bitte aber verzichten.