# Teilen binär - Vorzeichenlose Zahlen

Irgendwann steht jeder Programmieren vor diesem Problem. Wie teile ich eine Binärzahl durch eine zweite?

Wer in der Grundschule ein wenig aufgepasst hat sollte in der Lage sein schriftlich eine Zahl durch eine zweite zu teilen.

Hier ein Beispiel:

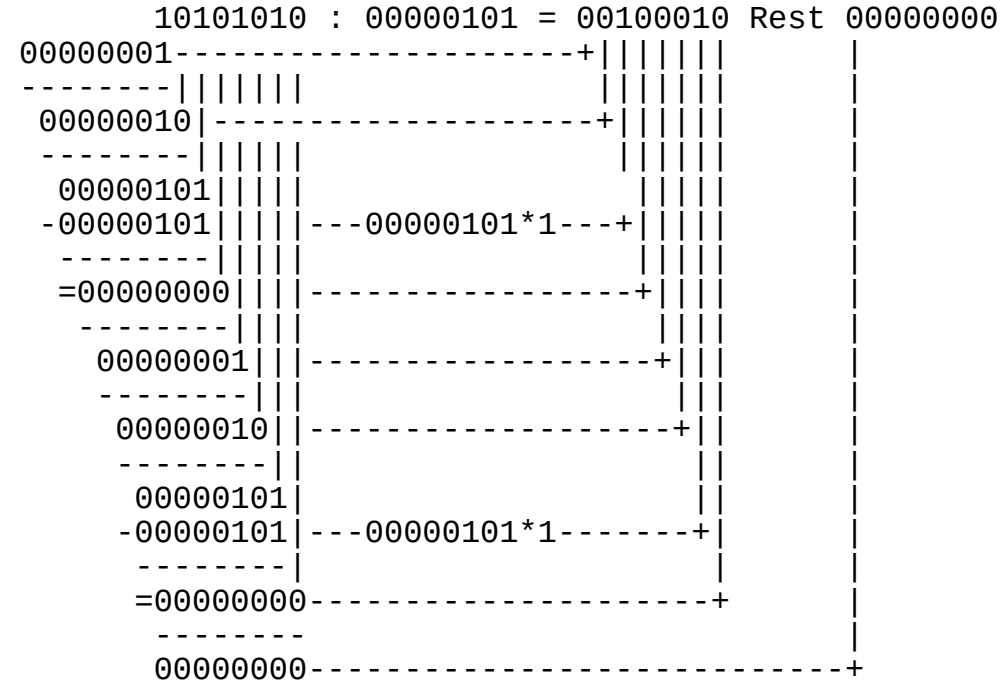
```
------------------
        170 : 5 = 34 Rest 0
       -15|--3*5--+|      |
        --|        |      |
        =20        |      |
        -20 --4*5--+      |
         --               |
         =0 --------------+
```

Nun. Da die Mathematik so universell ist sollte das ganze in Binärdarstellung doch genauso gehen. Probieren wir's mal aus:

```
        170 dez. = 10101010 bin.
          5 dez. =      101 bin.
       ( 34 dez. =   100010 bin.)

        10101010 : 101 = 100010 Rest 0
       -101|------101*1--+|||||      |
        ---|||||          |||||      |
         =00|-------------+||||      |
          --||||           ||||      |
           01--------------+|||      |
           --|||            |||      |
            10-------------+||      |
            --||            ||      |
            101|            ||      |
           -101----101*1-----+|      |
            ---|              |      |
             =00-------------+      |
               --                   |
               =0---------------------+
```

Die Sache klappt tatsächlich. Aber DAS in µP-Code um zu setzen scheint doch eine recht vertrackte Sache. Hier 8 Bit Divident, da drei Bit Divisor, dort 6 Bit Ergebnis und letztlich ein Bit Rest. Wie würde die Sache denn aussehen, wenn ich nur mit 8 Bit das Ganze wiederhole?




Scheinbar klappt das genau so. Aber wie beschreibe ich das so, dass man das auch auf einem µP programmieren kann?

1.) Ich habe dafür genau 8 Einzelschritte benötigt. Also müsste ich wohl eine Schleife bauen.

2.) Ich habe mit dem hochtwertigsten Bit 7 begonnen. Dies habe ich in ein Hilfsbyte an die niederwertigste Stelle kopiert.

3.) Dann habe ich geprüft, ob Das Hilfsbyte größer oder gleich dem Devisor war. Dem war nicht so und ich habe eine "0" an das hochwertigste Bit des Ergebnisses geschrieben.

4.) Das Hilfsbyte habe ich um ein Bit nach links geschoben und das Bit 6 des Dividenten hinten angehängt. Wieder wie vor geprüft und nochmals eine "0", diesmal an Bit 6, in das Ergebnis gesetzt.

5.) Wieder das Hilfsbyte geschoben, das nächste Bit angehängt und geprüft. Diesmal stand im Hilfsregister 5 und ich habe davon den Devisor 5 abziehen können und diesmal eine "1" im Ergebnis verhaftet.

6.) So habe ich also über alle 8 Bit immer gleich verfahren.

7.) Zuletzt blieb "0" im Hilfsregister. Dies musste der Rest der Division sein. Also hätte ich statt des Hilfsbytes gleich das Byte für den Rest nehmen können.

Bei genauer Betrachtung stellt man fest, dass ich den Dividenten in zwei Schritten ins Hilfsregister Rest schieben

könnte. MSBit des Divident über ROL ins Carry, Carry über ROL in Rest.

Dann Rest prüfen und ggf Devisor davon abziehen. Wenn abgezogen werden konnte, eine "1" sonst eine "0" im Ergebnis verhaften.

Auch diese Verhaftung könnte ich mir erleichtern. Im ersten Schritt wurde das Bit 7 als Zielbit gesetzt, im letzten Bit 0. Ich hätte auch immer Bit 0 setzen können und auch jeweils vorher ROLieren können. Nach 8 Schritten käme ich auf das Gleiche raus.

Wenn ich also so verfahre, fällt mir eines auf. Ich habe den Dividenten verloren. Er ist am Ende "0". Unschön aber auch wiederum schön. Ich könnte doch dieses Register für das Ergebnis verwenden. Ich ROLiere es eh vor jedem Schritt. Und sinnvolle  Daten würden eh nicht darin zurück gegeben.

Wie sähe jetzt der ganze Ablauf aus?

1.) Lösche Register REST

2.) Erzeuge Schleife über 8 Bit

  3.) Sicherer: Clear CARRY (CLC)

  4.) ROLiere Bit 7 vom Divident ins CARRY.
      BIT 0 wird dabei "0"

  5.) ROLiere Carry an Bit 0 von Rest

  6.) Rest >= Divisor ?
        JA   - Subtrahiere Divisor von Rest,
               Setzte Bit 0 von Divident auf "1" (INC)
        NEIN - tue nichts.

7.) Bearbeite Schleifenzähler und springe ggf. weiter in
    Schleife.

War doch eigentlich ganz einfach.

Nun gut, ist nur 8 durch 8 Bit. Aber für viele Anwendungen reicht das schon.

Aber 16 durch 8 ist nicht viel aufwendiger.

1.) Lösche Register REST

2.) Erzeuge Schleife über 16 Bit

  3.) Sicherer: Clear CARRY (CLC)

4.) ROLiere Bit 7 vom Divident-LOW ins CARRY.
    BIT 0 wird dabei "0"
    ROLiere Bit 7 von Divident-HIGH ins CARRY.
    Bit 0 wird hierbei Bit 7 von Divident_LOW.

5.) ROLiere Carry an Bit 0 von Rest

6.) Rest >= Divisor ?
      JA   - Subtrahiere Divisor von Rest,
             Setzte Bit 0 von Divident auf "1" (INC)
      NEIN - tue nichts.

7.) Bearbeite Schleifenzähler und springe ggf. weiter in
    Schleife.

Das sollte meist genügen. Meist soll eh nur ein Wort vom A/D-Wandler für die LCD-Anzeige umgerechnet werden. Dafür reicht 16div8. Teile durch 10, Rest ist BCD einer Stelle. Rest + 49 ergibt ASCII-Code. Raus auf's Display.

Wie das Ganze mit Vorzeichenbehafteten Zahlen aussieht und wie man multipliziert in einem anderen Beitrag.

Viel Erfolg erstmal mit der Codeumsetzung hierzu.