Multiplizieren binär - Vorzeichenlose Zahlen

Irgendwann steht jeder Programmieren vor diesem Problem.
Wie multipliziere ich eine Binärzahl mit einer zweiten?

Wer in der Grundschule ein wenig aufgepasst hat sollte in der Lage sein schriftlich eine Zahl mit einer zweiten zu multiplizieren.

Hier ein Beispiel:

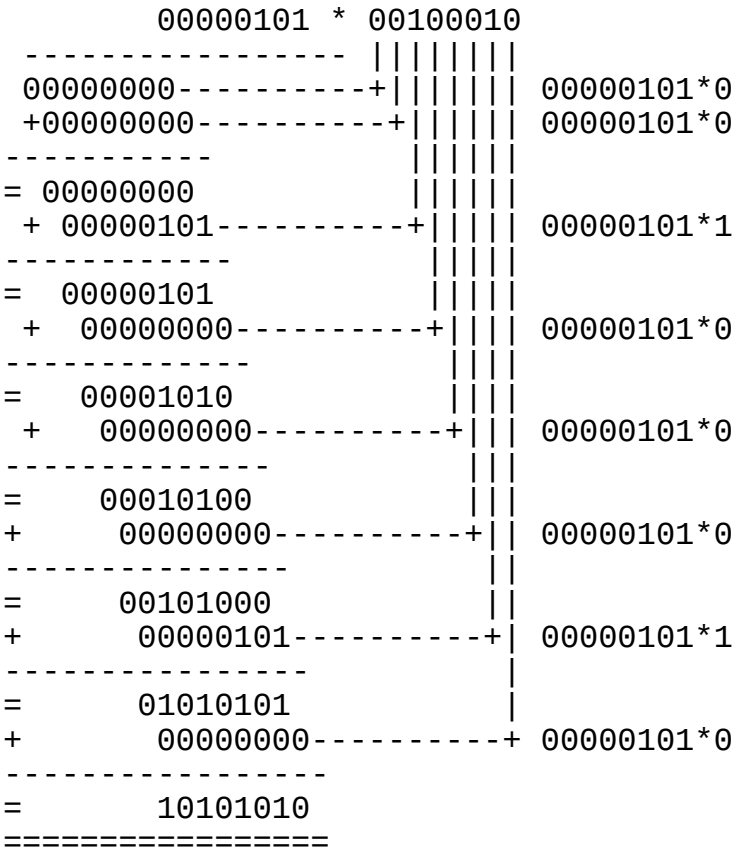
```
------------------
        5 * 34
       -----||
       15---+|  3*5
      + 20---+  4*5
      ---------
      =170
      =====
```

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

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

        101 * 100010
       -------||||||
        101---+|||||  101*1
         000---+||||  101*0
          000---+|||  101*0
           000---+||  101*0
            101---+|  101*1
       +     000---+  101*0
      ----------------------
       =10101010
      ===========
```

Die Sache klappt tatsächlich. Aber DAS in µP-Code um zu setzen scheint doch eine recht vertrackte Sache. Hier 3 Bit Multiplikant, da 5 Bit Multiplikator, dort 8 Bit Ergebnis.
Wie würde die Sache denn aussehen, wenn ich nur mit 8 Bit das Ganze wiederhole? Mal ganz ausführlich.



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 des Multiplikator begonnen. Dieses war "0" und ich habe Quasi eine "0" ins Ergebnissregister geschrieben.

3.) Dann habe ich Bit 6 des Multiplikator genommen. Das war auch "0" und ich habe "0" auf das Ergebnisregister addiert, dieses aber vorher quasi um ein Bit verschoben.

4.) Bit 5 des Multiplikators war "1" und ich habe nun den Multiplikant auf das wiederum verschobene Ergebnisregister addiert.

5.) So habe ich also über alle 8 Bit immer gleich verfahren. Ergebnis verschieben. Nichts oder den Multiplikant auf das Ergebnis addiert.

6.) Irgendwie habe ich dabei aber immer irgendwie Bits aus dem Ergebnis raus geschoben. Trotzdem stimmt es noch.

Bei genauer Betrachtung stellt man fest, dass hier eigentlich

ein 16 Bit Wert hätte herauskommen müssen. Es passierte aber nicht, da bei dem Beispiel nur ein 8Bit-Ergebnis heraus kam. Probieren wir doch die Rechnung nochmal mit anderen Zahlen. 200 * 200 = 40000 scheint leicht zu sein.

```
       200 dez. = 11001000 bin.
     (40000 dez. = 1001110001000000 bin.)

              11001000 * 11001000
          ----------------- ||||||||
          11001000----------+||||||| = 11001000 * 1
      +    11001000----------+|||||| = 11001000 * 1
        -----------           ||||||
      = 1001011000            ||||||
      +    00000000----------+|||||  = 11001000 * 0
        ------------           |||||
      = 10010110000            |||||
      +     00000000----------+||||  = 11001000 * 0
        -------------           ||||
      = 100101100000            ||||
      +      11001000----------+|||  = 11001000 * 1
        --------------           |||
      = 1001110001000            |||
      +       00000000----------+||  = 11001000 * 0
        ---------------           ||
      = 10011100010000            ||
      +        00000000----------+|  = 11001000 * 0
        ----------------           |
      = 100111000100000            |
      +         00000000----------+  = 11001000 * 0
        -----------------
      = 1001110001000000
      ==================
```

Das scheint doch eher der Wirklichkeit gerecht zu werden. Also überarbeite ich mal meine Gedankenschritte:

1.) Ich brauche 16 Bit für das Ergebnis.

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

3.) Ich habe mit dem hochtwertigsten Bit 7 des Multiplikator begonnen. Dieses war "1" und ich habe den Multiplikator scheinbar ins High-Byte des Ergebnissregister geschrieben.

4.) Dann habe ich Bit 6 des Multiplikator genommen. Das war auch "1" und ich habe den Multiplikator auf das Ergebnis-register addiert, dieses aber vorher quasi um ein Bit verschoben.

5.) Bit 5 des Multiplikators war "0" und ich habe nun Null

auf das wiederum verschobene Ergebnisregister addiert.

6.) So habe ich also über alle 8 Bit immer gleich verfahren.
Ergebnis verschieben. Nichts oder den Multiplikant auf das Ergebnis addiert.

7.) Es gingen keine Bits bei der Schieberei verloren.

Wie schon in "Teilen binär" scheint alles irgendwie auf Basis Register schieben heraus zu laufen.

Wiederum spielt es im Prinzip keine Rolle ob ich im dritten Schritt den Multiplikator ins High-Byte schreibe oder auf das Low-Byte Addiere und dann Low über High schiebe. Ganz im Gegenteil. Es vereinfacht die Sache ungemein.

Auch hier geht bei der Bearbeitung ein Register, der Multiplikator, verloren und kann registersparend als High-Byte des Ergebnisses verwendet werden. Aber Achtung: Dann darf Schritt 6 des folgenden Ablaufplanes nicht ausgeführt werden. Das wäre in Schritt 5 schon erfolgt, da es ja das selbe Register ist. Klar?

Wie sähe jetzt der ganze Ablauf aus?

1.) Lösche Ergebnisregister-Low.
High braucht nicht gelöscht werden. Inhalt geht bei der folgenden Schieberei eh über die Wupper.

2.) Erzeuge Schleife über 8 Bit

  3.) Sicherer: Clear CARRY (CLC)

  4.) ROLiere Bit 7 vom Ergebnis-Low ins CARRY.
  BIT 0 wird dabei "0"

  5.) ROLiere Carry an Bit 0 von Ergebnis-High

  6.) ROLiere Bit 7 von Multiplikator ins CARRY

  7.) CARRY gesetzt ?
  JA - Addiere Multiplikant auf Ergebnis-Low,
  NEIN - tue nichts.

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

War doch eigentlich auch ganz einfach.

Nun gut, ist nur 8 mal 8 Bit auf 16. Aber für viele Anwendungen reicht das schon.

Viel Erfolg erstmal mit der Codeumsetzung hierzu.