Binäre Division mit Rest

Arithmetic Logic Unit (ALU 9+8 Bit)

```
    Bit 1                  Bit
    Vorzeichenerweiterung  123456789       fta
                                           000          Steuerbits fta
                                     00000000 ALS      Schiebezeiger
        Operand 1     AL1 000000000  00000000 ALD      Dividend Schieberegister
        Operand 1     AL2 000000000  00000000 ALN      negativer Divisor
        Übertrag      ALC 000000000  00000000 ALP      positiver Divisor
        Rest bei DIV  ALR 000000000  00000000 ARL      Resultat bei DIV
     (bei add Resultat)       9 Bit      8 Bit       (1+16Bit Resultat Low Byte)
```

Ablaufsteuerung mit Steuerbits: f (first loop) t (two) a (add)

f: Erster Gesamtdurchlauf im Bitvergleich von Dividend in AL1 und Divisor in AL2 (f=1).
   Der Dividend wird bei Bedarf nach rechts in das Hilfsregister ALD des Dividenden verschoben.
   Im Schiebezeiger Register ALS wird von links je eine 1 eingeschoben.
t: Im ersten Teilablauf werden von links die Bits verglichen, eventuell wird geschoben.
   Ergibt sich daraus, dass zur Größenbestimung auch die Folgebits im zweiten Teilablauf geprüft werden müssen,
   wird t=1 gesetzt.
a: Lösungsbit 1 wird an den Quotienten angehangen, dann Addition des negativen Divisors (aktueller Rest),
   ein Bit aus dem Hilfsregister ALD des Dividenden wird nachgeschoben.
   Dies erfolgt nur, wenn im darüberstehenden Schiebezeiger ALS links noch eine 1 steht.
   Die Addition (add) erfolgt mit den 9 Bit Registern.

Der Divisions-Algoritmus - für positive Operanden -

```
    set ALN negativer Divisor          // vorab zu berechnen, 0 ist vorher abzufangen
    set ALP Divisor                    // 1 ist vorher abzufangen, x\1 = x Rest 0
    set ALS 00000000                   // Schiebezeiger initialisieren
    set ALD 00000000                   // Schieberegister Dividend initialisieren
    set ARL 00000000                   // Resultat (Quotient) initialisieren
    set ALf 1                          // Vorbelegung  erster Gesamtdurchlauf (äußere Schleife)
    set ALt 0                          // Vorbelegung keine zweiter Teilablauf (Teilschleife)
    set ALa 0                          // Vorbelegung keine Addition des negativen Divisors
    set AL1 0+Dividend                 // mit Vorzeichenerweiterung 0
    set AL2 0+Divisor                  // mit Vorzeichenerweiterung 0
    // ------------------------------------------------------------------------------
    // Hauptschleife, Abbruch nur mit break,  Teil-Dividend über Divisor platzieren
    loop
      bit = 3     // Bits AL1, AL2 von links prüfen, ohne Vorzeichenerweiterung Bit 1 und Vorzeichenbit 2 (0 positiv)
      // erste Teilschleife,  Abbruch nur mit break
      loop
        b1 = get(AL1,bit)                              // aktuelles Bit auslesen
        b2 = get(AL2,bit)                              // aktuelles Bit auslesen
        bs = getleft(ALS,1)                            // linkes Bit Schiebezähler  (bs=1, links schieben möglich)
        // ---------------
        if b1=0 ∧ b2=0
          if bit=9
            // bei bit=9  wäre Division 0\0 nicht definiert, vorher abzufangen!
          else
            increment bit                              // vorlaufende Nullen übergehen
          endif // bit
        endif
        // ---------------
```

```
  // ---------------
  if b1=0 ∧ b2=1
    if bs=1    // schieben möglich,  links schieben, ALS mitschieben (AL1 9 Bit beachten)
      set AL1 getright(AL1,8)+getleft(ALD,1)         //   << AL1_ALD(1)  (Bitzähler nicht erhöhen!)
      set ALD getright(ALD,7)+0                      //   << ALD(2..8)_0
      set ALS getright(ALS,7)+0                      //   Schiebezähler links schieben, 0 anhängen
      set ARL getright(ARL,7)+0                      //   Lösungs-Bit 0 anhängen (für Stelle links schieben)
    else       //  kein links schieben mehr möglich,  Dividend < Divisor
      set ALt 0                                      // keine zweite Schleife,
      set ALa 0                                      // keine Addition negativer Divisor
      set ALR AL1                                    // aktueller Rest aus AL1 nach ALR
      set ARL getright(ARL,7)+0                      // Lösungs-Bit 0 anhängen
      break
    endif // bs
  endif
  // ---------------
  if b1=1 ∧ b2=0
    if bit=9
      // bei bit=9  wäre Division 1\0 nicht definiert, vorher abzufangen!
    else
      if ALf=1     // nur beim ersten Durchlauf in äußerer Schleife: linke Position Dividend zu Divisor
        set ALD getright(AL1,1)+getleft(ALD,7)       //   >> AL1(9)_ALD
        set AL1 0 + getleft(AL1,8)                   //   >> 0_AL1(1..8)
        set ALS 1 + getleft(ALS,7)                   // Schiebezähler mit 1 von links füllen
        increment bit
      else
        set ALt 0                                    // keine zweite Schleife
        set ALa 1                                    // Addition negative Divisor
        set ARL getright(ARL,7)+1                    // Lösungs-Bit 1 anhängen
        break
      endif // ALf
    endif // bit
  endif
  // ---------------
  if b1=1 ∧ b2=1
    if bit=9
      // bei bit=9  wäre Division  1\1 = 1  Rest 0   Division durch 1 vorher abzufangen!
    else
      set ALt 1                                      // zweite Schleife (da 1 über 1 gefunden)
      increment bit                                  //        ... ab Folgebit
      break
    endif // bit
  endif
  //----------------
end_loop // erste Teilschleife
//***************************
```

```
if ALt=1                      //  <---  eventuell Teilschleife zwei durchlaufen
//**************************
  loop  // zweite Teilschleife, Abbruch nur mit break, weitere Bits nach rechts prüfen
    b1 = get(AL1,bit)                                 // aktuelles Bit auslesen
    b2 = get(AL2,bit)                                 // aktuelles Bit auslesen
    bs = getleft(ALS)                                 // Schiebezähler linkes Bit (1 schieben möglich)
    // ---------------
    if b1=0 ∧ b2=0
      if bit=9
        if bs=1              // nur Zwischenergebnis // weiterhin noch schieben möglich
          set ALa 1                                   // Addition negativer Divisor
        else //   letzte Teilrechnung
          set ALa 0                                   // keine Addition negativer Divisor
          set ALR 000000000                           // Rest 0 nach ALR
        endif // bs
        set ARL getright(ARL,7)+1                     // Lösungs-Bit  anhängen
        break
      else
         increment bit
      endif // bit
    endif
    // ---------------
    if b1=0 ∧ b2=1
      if bs=1     // links schieben möglich, ALS mitschieben, (AL1 9 Bit beachten)
        set AL1 getright(AL1,8)+getleft(ALD,1)        //   << AL1_ALD(1) (Bitzähler nicht erhöhen!)
        set ALD getright(ALD,7)+0                     //   << ALD(2..8)_0
        set ALS getright(ALS,7)+0                     // Schiebezähler links schieben, 0 anhängen
        set ARL getright(ARL,7)+0                     // Lösungs-Bit 0 anhängen (für Stelle links schieben)
        set ARL getright(ARL,7)+1                     // Lösungs-Bit 1 anhängen (für 'Division')
        set ALa 1                                     // Addition negativer Divisor, Position erreicht
        break
      else
        set ALa 0                                     // kein schieben möglich, keine Addition negativer Divisor
        set ARL getright(ARL,7)+0                     // Lösungs-Bit 0 anhängen
        set ALR AL1                                   // Rest nach ALR
        break
      endif // bs
    endif
    // ---------------
    if b1=1 ∧ b2=0
      set ARL getright(ARL,7)+1                       // Lösungs-Bit 1 anhängen
      set ALa 1                                       // Addition negativer Divisor, Position erreicht
      break                                           // Dividend größer Divisor
    endif
    // ---------------
```

```
      // ---------------
      if b1=1 ∧ b2=1
        if bit=9
          if bs=1              // nur Zwischenergebnis // weiterhin noch schieben möglich
            set ALa 1                                 // Addition negativer Divisor
          else //   letzte Teilrechnung
            set ALa 0                                 // keine Addition negativer Divisor
            set ALR 000000000                         // Rest Null nach ALR
          endif // bs
          set ARL getright(ARL,7)+1                   // Lösungs-Bit anhängen
          break
        else
          increment bit
        endif // bit
      endif
      // ---------------
    end_loop // zweite Teilschleife
    // ***************************
  endif  // ALt
  // ================================================
  //
  if ALa=1                    //  <---  eventuell Addition negativer Divisor
    //
    set AL1 getright(AL1,8)                           // Vorzeichenerweiterung erfolgt bei Addition
    set AL2 ALN                                       // negativer Divisor nach AL2 zur Addition
    // Addition in der ALU ausführen, das Vorzeichenbit von AL1 und AL2 wird vorab dupliziert.
    //
    add   //  <-- Additions-Algoritmus add_ALU_9  (AL1+AL2) im 9 Bit-Rechenwerk mit Vorzeichenerweiterung
    //
    bs = getleft(ALS,1)                               // Schiebezähler linkes Bit
    if bs=1                   // Schiebezähler linkes Bit ist 1, Bit des Dividenden nachschieben möglich
      set AL1 ALR                                     // Differenz in ALR ist Rest,  wieder in AL1 setzen
      set AL2 0+ALP                                   // positiven Divisor mit Vorzeichenerw. in AL2 setzen
      //                          ... weiteres Bit des Dividenden nachschieben
      set AL1 getright(AL1,8)+getleft(ALD,1)          //   << AL1_ALD(1)  (Bitzähler nicht erhöhen!)
      set ALD getright(ALD,7)+0                       //   << ALD(2..8)_0
      set ALS getright(ALS,7)+0                       // Schiebezähler links schieben, 0 anhängen
    else
      break                                           // fertig,  Rest in ALR
    endif // bs
  else //  keine Addition negativer Divisor
    break                                             // Rest in ALR
  endif // ALa
  //================================================
  if  ALf=1
    set ALf 0          // Kennung 0 für weitere Durchläufe der äußeren Schleife nach erstem Durchlauf
  endif // ALf
end_loop // äußere Schleife
//
//Rest in ALR,  Quotient in ARL
```

```
// HINWEISE:   Einschränkung des Zahlenbereichs -128 .. 127 dezimal,
//             -128 ist als Operand nicht möglich und führt bei Bildung des Zweierkomplements zum Überlauf.
//             Die Dezimalzahlen 0 und 1 als Divisor sind vorher abzufangen.
```