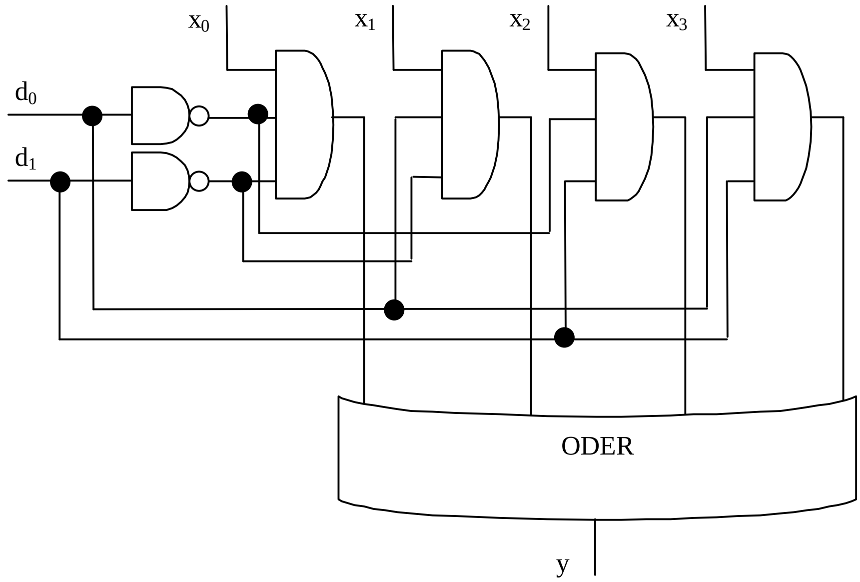
Aufgabe 37



Aufgabe 38

a) 1637 – 101 = 1637 + (-101)
1637 = 0000 0110 0110 0101, 101 = 0000 0000 0110 0101

Im Einerkomplement:	1637		0000 0110 0110 0101
	-101		1111 1111 1001 1010
		1	0000 0101 1111 1111
			+1
			0000 0110 0000 0000 = 1637-101=1536

b) Die finale Addition des Carry bedeutet, dass zur bisher gebildeten Summe noch die Zahl 1 addiert wird. Ein weiterer Übertrag kann nur entstehen, wenn die bisherige Summe die Form 111...111 hat. Diese Ziffernfolge entspricht jedoch der Zahl –0, die identisch mit +0 = 000...000 ist. Wenn man sich diese Beziehung zunutze macht und sofort die Umformung durchführt, kann die nachfolgende Addition des Carry keinen Überlauf mehr erzeugen.

Aufgabe 39

a) Der Booth-Algorithmus führt die Multiplikation von Binärzahlen auf einfache Bitverschiebungen und Additionen zurück. Dazu wird einer der beiden Faktoren in eine Summe aus positiven und negativen Zweierpotenzen nach einem vorgegebenem Schema zerlegt und dann, je nach Vorzeichen, nach Bitverschiebungen aufaddiert.

b) Bei jeder Verschiebung um n Bits werden n Bits auf einer Seite herausgeschoben und auf der anderen müssen n Bits eingefügt werden. Logische Shifts setzen die neuen Bits stets auf Null. Bei arithmetischen Rechtsshifts wird dagegen das alte MSB repliziert, um die Zweierkomplementdarstellung zu wahren.

c) 0100 * 0111: 0100 = 2^3-2^2

	0000
arith.sh.	00000
arith.sh.	000000
-2^2	0111
K_2	1001 +
	100100
arith.sh.	1100100
+2^3	0111 +
1	0011100 = 28

Aufgabe 40

a) Die Division von Binärzahlen ähnelt dem Algorithmus für Dezimalzahlen.
In jedem Schritt sind folgende Überlegungen durchzuführen:
Ist die führende Ziffer eine 1 ?

- ja, subtrahiere 1x den Divisor (Zweierkomplement beachten !)
  Ist das Resultat positiv ?
  - ja, 1 im Ergebnis bestätigt, shifte Divisor eine Stelle nach rechts
  - nein, 1 im Ergebnis durch 0 ersetzen, addiere 1x Divisor, shifte Divisor eine Stelle nach rechts
- nein, shifte Divisor eine Stelle nach rechts
  ergänze 0 im Ergebnis

Bezogen auf die Beispielrechnung ergibt sich:

$K_2(00001001) = 11110111$

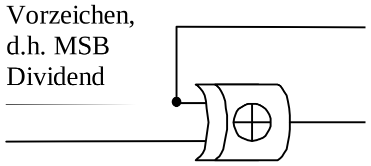
```
00101010 : 00001001 = 001
 (1001)
 +0111
1 000110
```

b) Der absolute Betrag des Quotienten und des Restes ist unabhängig vom Vorzeichen. Demzufolge führt man zuerst die vorzeichenlose Division durch und bestimmt danach die Vorzeichen des Quotienten und des Restes nach folgendem Schema:

Dividend	Divisor	Quotient	Rest
+	+	+	+
-	+	-	-
+	-	-	+
-	-	+	-

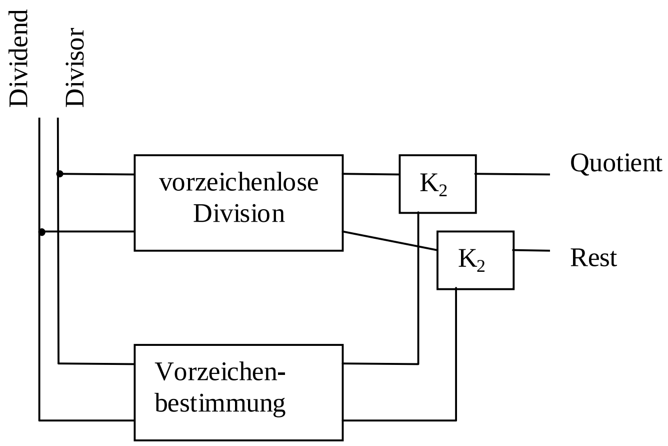
c) Beim Dividierer sind zwei grundsätzliche Funktionseinheiten zu unterscheiden (Bitoperationen in C- bzw. Java-Kurzschreibweise):
1. Dividierer für positive Zahlen

2. Ermittlung des Vorzeichens der Ergebnisse für positive und negative Operanden



Nachdem das Vorzeichen bestimmt wurde, muss noch eventuell das Zweierkomplement K_2 des Restes bzw. des Quotienten gebildet werden.

Diese beiden Einheiten bilden zusammen das Divisionswerk:



Die K_2-Einheiten sind als Multiplexer zu verstehen, die, wenn der Eingang, der der Vorzeichenbestimmung entstammt, 1 ist, das Zweierkomplement des anderen Eingangs, der ein Teil des Divisionsergebnisses ist, bilden.