Triangle Technology Executive Panel

Learn what companies are looking for in employees

Friday, 2 March
3:30 p.m.
Sitterson 014

Network

For more info, see
http://www.cs.unc.edu/~pozefsky/TTECPanel.html

Find out about local businesses

Ask questions about the industry

Capital Analytics

Help with Pete's user study

Interested in mobile audio?

Help us evaluate an auditory computer interface!

All participants will be paid $20.00 upon completion of the study.

http://www.cs.unc.edu/~parente/clique/sighted-eval.html

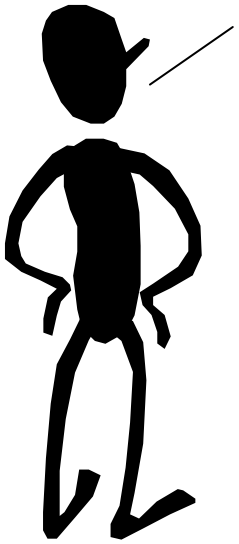
Contact: Peter Parente <parente@cs.unc.edu>

Binary Multipliers

The key trick of multiplication is memorizing a digit-to-digit table…
Everything else was just adding

×	0	1	2	3	4	5	6	7	8	9
0	0	0	0	0	0	0	0	0	0	0
1	0	1	2	3	4	5	6	7	8	9
2	0	2	4	6	8	10	12	14	16	18
3	0	3	6	9	12	15	18	21	24	27
4	0	4	8	12	16	20	24	28	32	36
5	0	5	10	15	20	25	30	35	40	45
6	0	6	12	18	24	30	36	42	48	54
7	0	7	14	21	28	35	42	49	56	63
8	0	8	16	24	32	40	48	56	64	72
9	0	9	18	27	36	45	54	63	72	81

×	0	1
0	0	0
1	0	1



You've got to be kidding… It can't be that easy

Reading: Study Chapter 3.

Binary Multiplication

The "Binary" Multiplication Table

X	0	1
0	0	0
1	0	1

Binary multiplication is implemented using the same basic longhand algorithm that you learned in grade school.

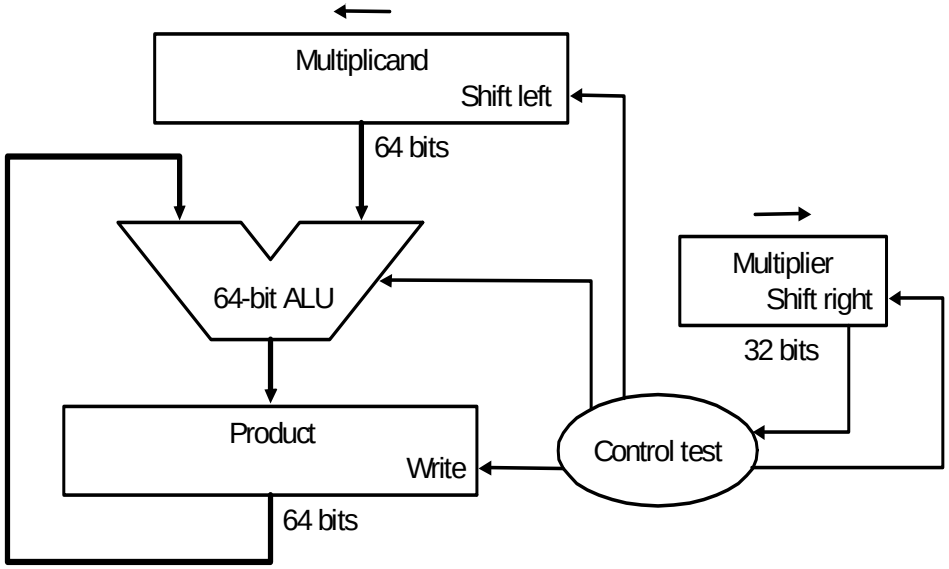
$$
\begin{array}{rcccccc}
 & & A_3 & A_2 & A_1 & A_0 \\
\times & & B_3 & B_2 & B_1 & B_0 \\
\hline
 & & A_3B_0 & A_2B_0 & A_1B_0 & A_0B_0 \\
 & A_3B_1 & A_2B_1 & A_1B_1 & A_0B_1 & \\
 & A_3B_2 & A_2B_2 & A_1B_2 & A_0B_2 & \\
+ & A_3B_3 & A_2B_3 & A_1B_3 & A_0B_3 & \\
\hline
\end{array}
$$

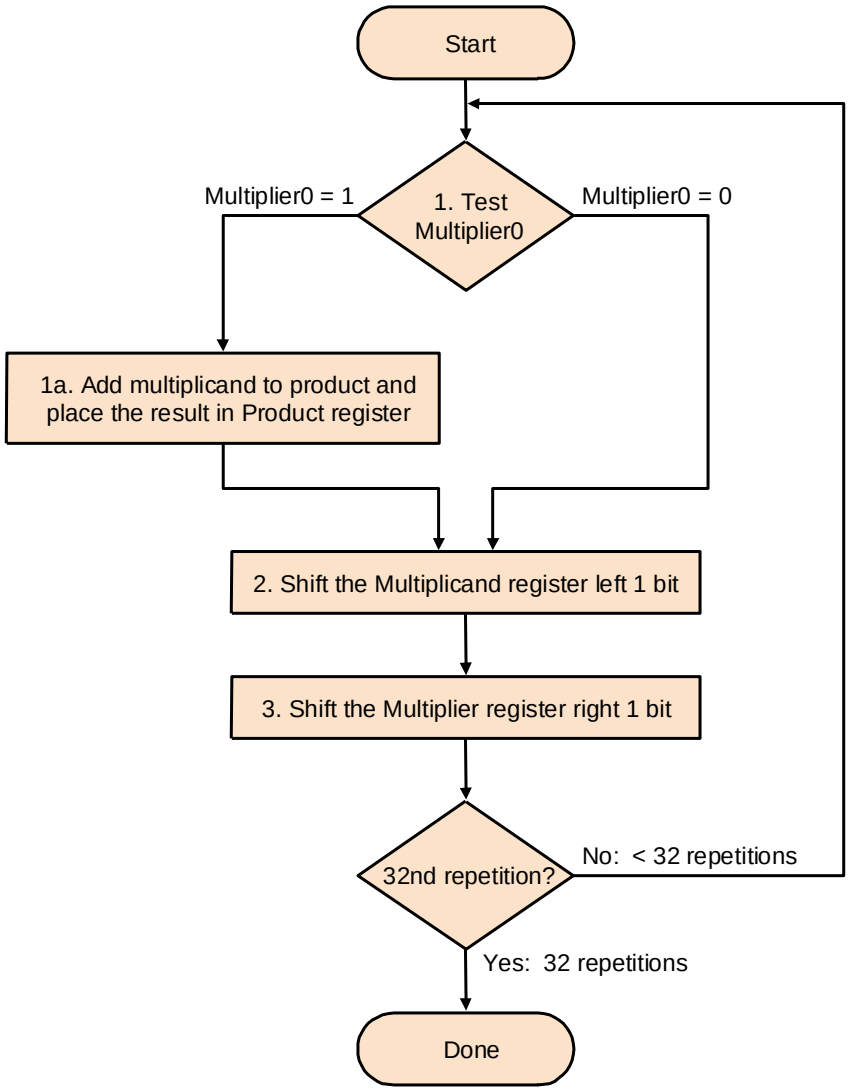
A_jB_i is a "partial product"

Multiplying N-digit number by M-digit number gives (N+M)-digit result

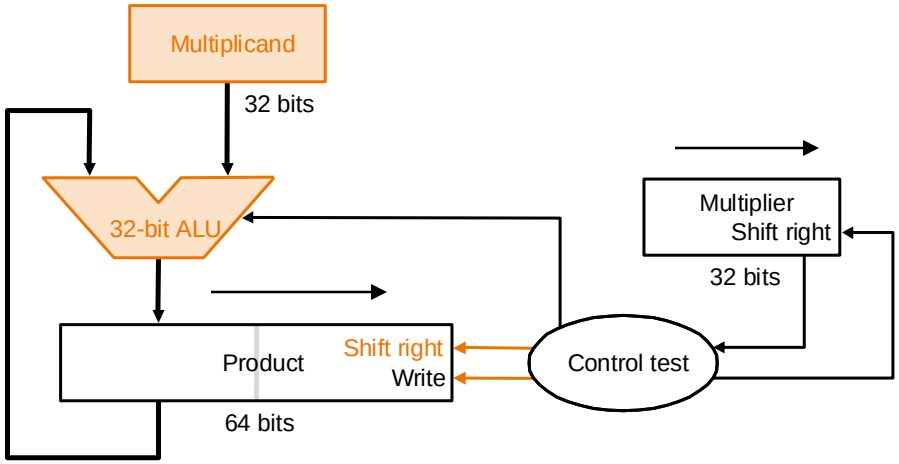
Easy part: forming partial products (just an AND gate since B_i is either 0 or 1)
Hard part: adding M, N-bit partial products

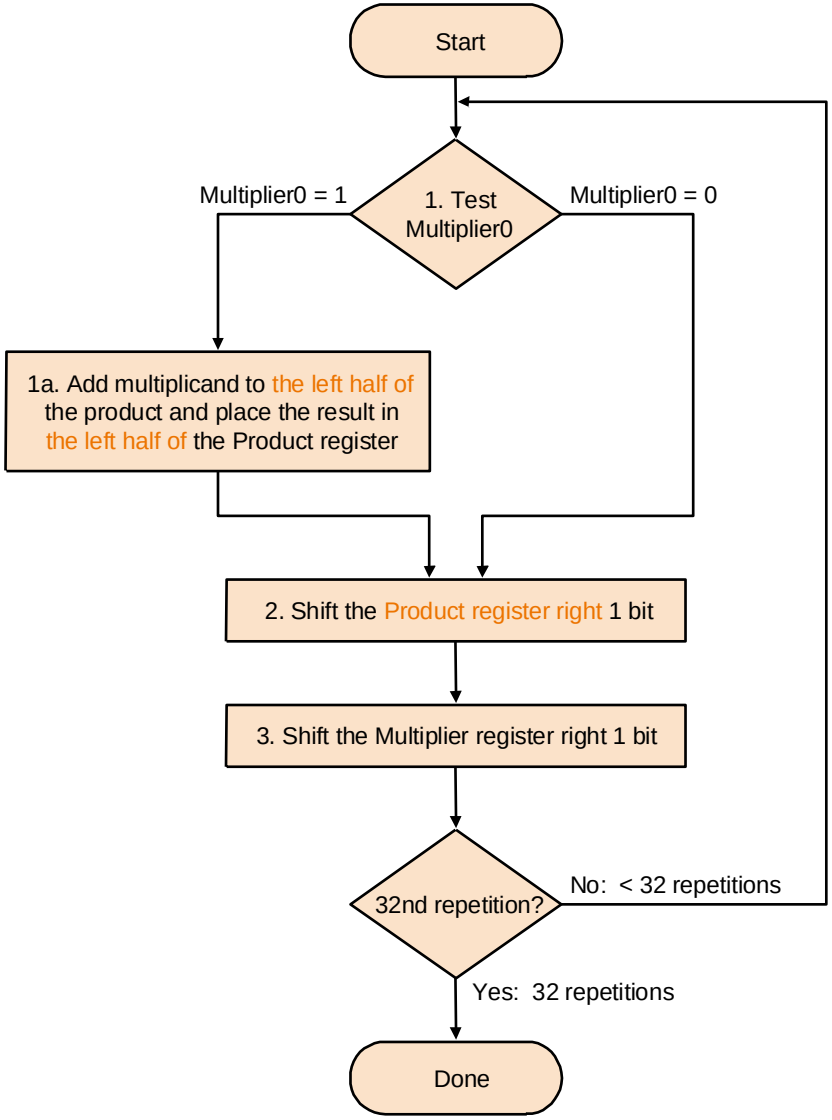
Multiplication: Implementation





Second Version

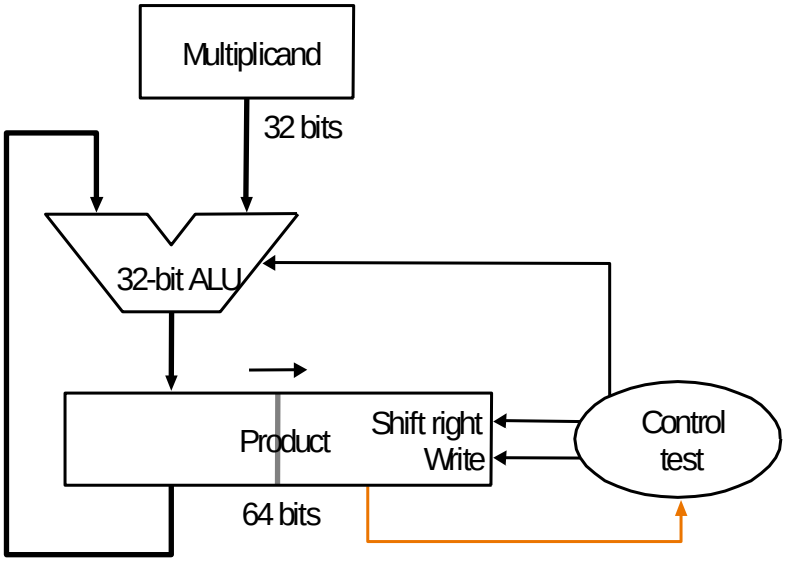


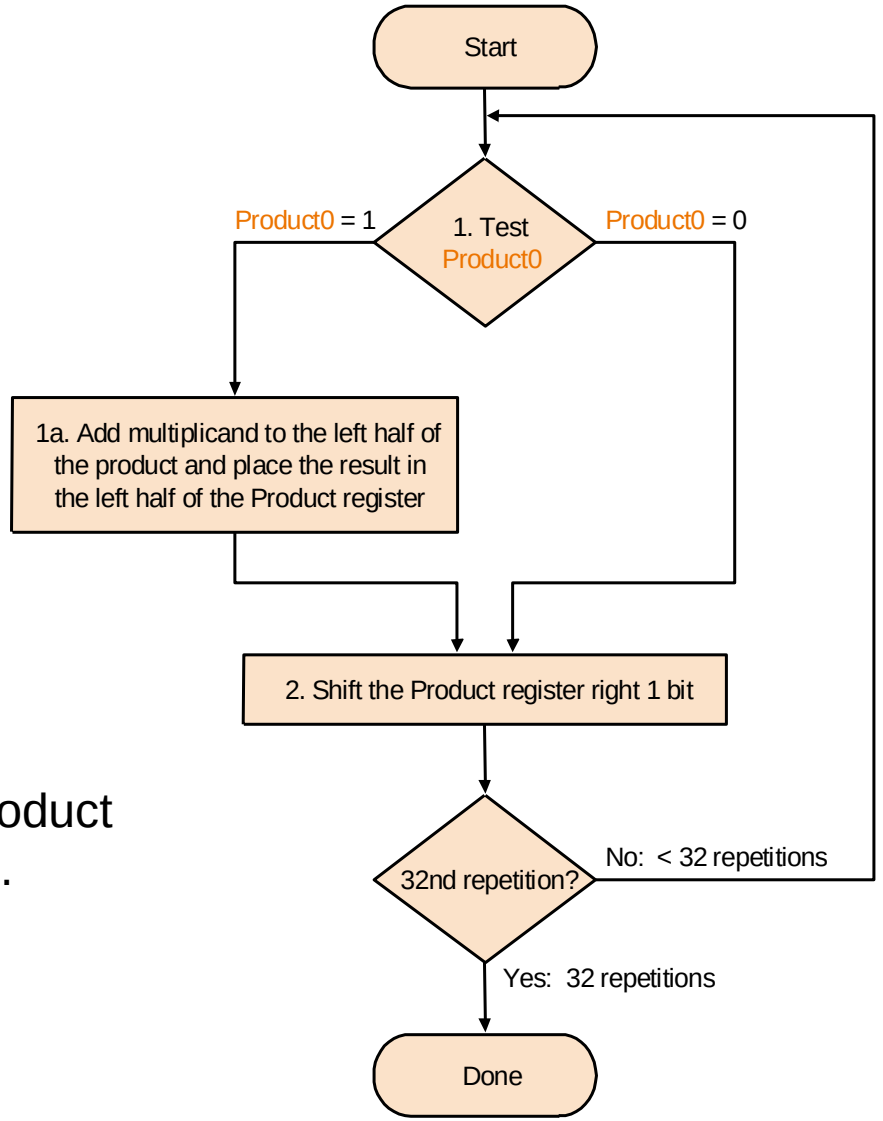


Example for second version

Iteration	Step	Multiplier	Multiplicand	Product
0	Initial	1011	0010	0000 0000
1	Test true shift right	1011 0101	0010	0010 0000 0001 0000
2	Test true shift right	0101 0010	0010	0011 0000 0001 1000
3	Test false shift right	0010 0001	0010	0001 1000 0000 1100
4	Test true shift right	0001 0000	0010	0010 1100 0001 0110

Final Version





The trick is to use the lower half of the product to hold the multiplier during the operation.

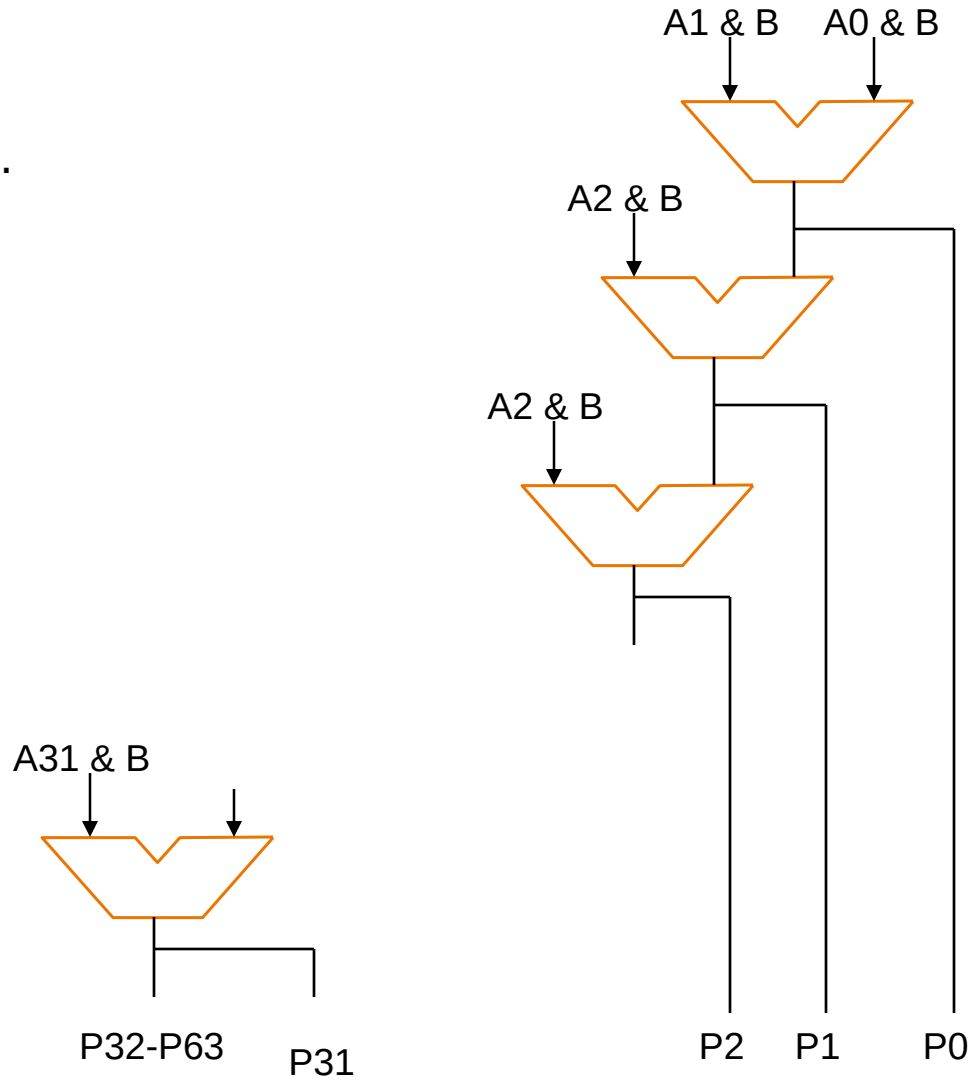
What about the sign?

Positive numbers are easy.

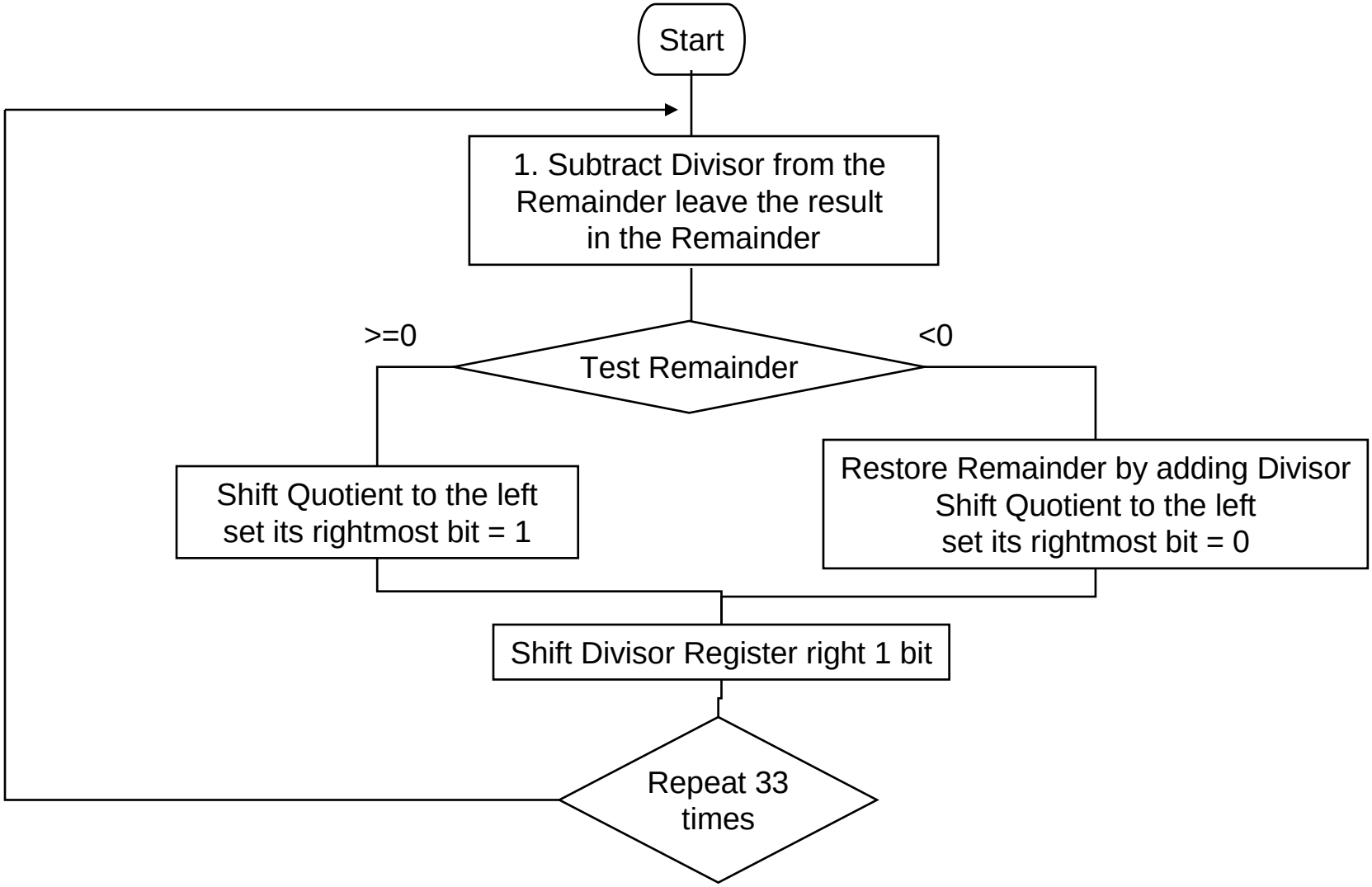
How about negative numbers?

Faster Multiply

See Booth coding in the book.



Division


Start
1. Subtract Divisor from the Remainder leave the result in the Remainder
Test Remainder
>=0
<0
Shift Quotient to the left set its rightmost bit = 1
Restore Remainder by adding Divisor
Shift Quotient to the left
set its rightmost bit = 0
Shift Divisor Register right 1 bit
Repeat 33 times