Exploring Binary

Binary Multiplication

By Rick Regan February 21st, 2012

This is the third of a four part series on "pencil and paper" binary arithmetic, which I'm writing as a supplement to my binary calculator. The first article discusses binary addition; the second article discusses binary subtraction; this article discusses binary multiplication.

```
      1011.01
  x     110.1
  -----------
       101101
      0000000
     10110100
    101101000
  -----------
  1001001.001
```

Example of Binary Multiplication

The pencil-and-paper method of binary multiplication is just like the pencil-and-paper method of decimal multiplication; the same algorithm applies, except binary numerals are manipulated instead. The way it works out though, binary multiplication is much simpler. The multiplier contains only 0s and 1s, so each multiplication step produces either zeros or a copy of the multiplicand. So binary multiplication is not multiplication at all — it's just repeated binary addition!

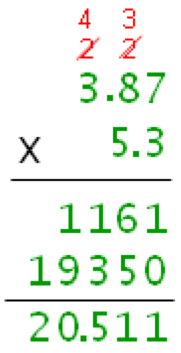
Decimal Multiplication

To multiply two multiple-digit decimal numbers, you first need to know how to multiply two single-digit decimal numbers. This requires the memorization of 100 facts, or 55 facts if you exclude the commutative or "turnaround" facts. These facts are usually represented in a "multiplication table," also known as a "times table." Example facts are 2 x 9 = 18, 9 x 7 = 63, and 1 x 6 = 6.

A multiplication problem is written with one number on top, called the multiplicand, and one number on the bottom, called the multiplier. The algorithm has two phases: the multiplication phase, where you produce what are called partial products, and the addition phase, where you add the partial products to get the result.

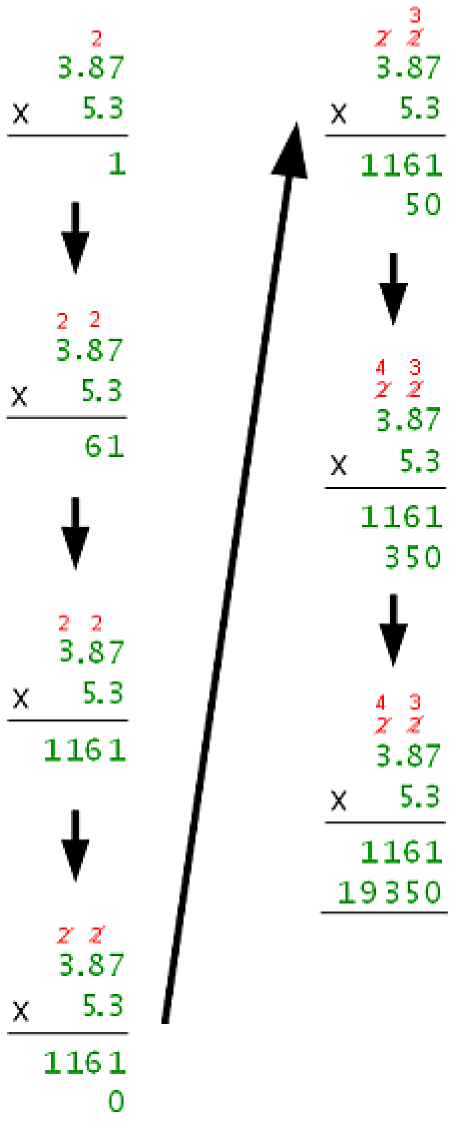
In the multiplication phase, the digits of the multiplier are stepped through one at a time, from right to left. Each digit of the multiplicand is then multiplied, in turn, by the current multiplier digit; taken together, these single-digit multiplications form a partial product. The answer to each single-digit multiplication comes from the multiplication table. Some of these answers are double-digit numbers, in which case the least significant digit is recorded and the most significant digit is carried over to be added to the result of the next single-digit multiplication.

For example, let's multiply 3.87 and 5.3:



Example of Decimal Multiplication

There are two digits in the multiplier, so there are two partial products: 1161 and 19350. Each partial product has its own set of carries, which are crossed out before computation of the next partial product. Here is the multiplication phase, broken down into steps:



Steps of Decimal Multiplication (Multiplication Phase Only)

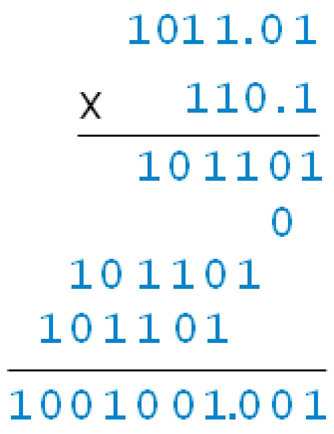
When the multiplication phase is done, the partial products are added, and the decimal point is placed appropriately. (If there were any minus signs, they would be taken into account at this point as well.) This gives the answer 20.511.

Binary Multiplication

Binary multiplication uses the same algorithm, but uses just three order-independent facts: 0 x 0 = 0, 1 x 0 = 0, and 1 x 1 = 1 (these work the same as in decimal). If you perform the multiplication phase with these facts, you'll notice two things: there are never any carries, and the partial products will either be zeros or a shifted copy of the multiplicand.

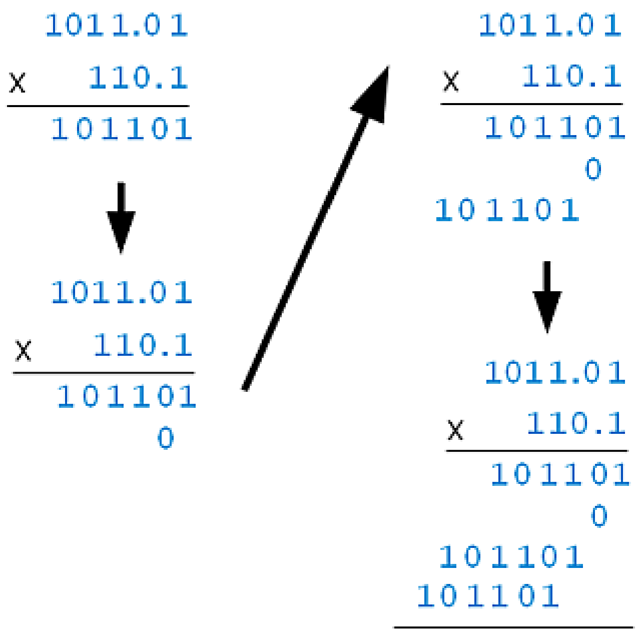
Observing this, you'll realize there's no need for digit-by-digit multiplication, which means there's no need to consult a times table — which means there's no multiplication, period! Instead, you just write down 0 when the current digit of the multiplier is 0, and you write down the multiplicand when the current digit of the multiplier is 1.

In the introduction, I showed this example: 1011.01 x 110.1. I wrote it as if you followed the decimal algorithm to the letter. Here's how it looks if you follow the simpler "write zero or multiplicand" algorithm (it's the same result, but with blanks representing 0s; this matches better conceptually with what we are now doing):



Example of Binary Multiplication (Simplified View)

Here's what the "multiplication" phase looks like, step-by-step:



Steps of Binary Multiplication (Multiplication Phase Only)

Each step is the placement of an entire partial product, unlike in decimal, where each step is a single-digit multiplication (and possible addition of a carry).

In the addition phase, the partial products are added using binary addition, and then the radix point is placed appropriately. This gives the answer 1001001.001.

Checking the Answer

You can check the answer by converting the operands to decimal, doing decimal multiplication, and then converting the decimal answer to binary. 1011.01 = 11.25, and 110.1 = 6.5. Their product is 73.125, which is 1001001.001, the answer we got using binary multiplication.

You can also check the answer using my binary calculator.

Discussion

Computers don't multiply in exactly this way, but they do exploit the simplified view of binary multiplication that I've described.

Related

- When Doubles Don't Behave Like Doubles
- When Floats Don't Behave Like Floats

Get articles by RSS (What Is RSS?)

Get articles by e-mail

Binary numbers / Binary arithmetic, Binary integers, Convert to binary, Convert to decimal

33 comments

1. **Shaun**

   March 14, 2012 at 1:55 am

   Hi Rick,

   Would you mind if I include this post in the Math and Multimedia Carnival #21, being hosted on my site, Math Concepts Explained at http://sk19math.blogspot.com. This is a great introduction to and demonstration of working with binary arithmetic!

   Cheers,
   Shaun

2. **Rick Regan**

   March 14, 2012 at 8:15 am

   Shaun,

   Sure, go ahead. I’m glad you found it useful.

   Rick

3. **Wilf**

   September 4, 2012 at 2:08 pm

Hi,

At the addition stage what happens when you add 1+1+1+1 for example?
v
10111100
10011101
10110010
10110010

---

4. **Rick Regan**

September 4, 2012 at 11:39 pm

@Wilf,

Think of it as three different addition problems: 1+1 = 10, 10 + 1 = 11, and 11 + 1 = 100. You might be able to do that last one in your head, but if not, just write it out:

```
11
 11
+ 1
---
100
```

---

5. **Oz**

November 24, 2013 at 2:35 pm

OK so .1 in binary is .5 in decimal, .01 is .25 and .001 is .125. Where do I learn how to display 3.87 and 5.3 in binary? Thank you.

---

6. **Rick Regan**

November 24, 2013 at 3:17 pm

@Oz,

I'm assuming you want the method that converts by hand. Separate the integer and fractional parts and convert them separately. For the integer part, use "repeated division by 2"; for the fractional part, use “repeated multiplication by 2” (search the Web for the details).

For example, convert 14.1 to binary. Start with the integer part:

14/2 = 7 remainder 0
7/2 = 3 remainder 1
3/2 = 1 remainder 1
1/2 = 0 remainder 1

The answer for the integer part is the remainders reversed: 1110.

Now do the fractional part:

0.1 * 2 = 0.2
0.2 * 2 = 0.4
0.4 * 2 = 0.8
0.8 * 2 = 1.6
0.6 * 2 = 1.2
0.2 * 2 = … (repeats)

The answer for the ~~integer~~ fractional part is 0.00011001100110011…

---

7. **Oz**

November 25, 2013 at 10:01 am

Thank you for your swift response! I had assumed operations in binary would be easy – and they are until one hits division. I can’t even handle 5/3. Thanks again.

---

8. **Asukele**

September 9, 2014 at 6:01 pm

It really helped me and I think I’ll be here more often as I go through college

---

9. **asif**

September 19, 2014 at 1:06 pm

tell me plz
how to find the position of decimal point in the result

---

10. **Rick Regan**

    September 19, 2014 at 7:57 pm

    @asif,

    Just add the number of places in both factors (the multiplier and multiplicand).

---

11. **swat**

    January 16, 2015 at 3:20 pm

    what are the four categories of binary multiplication?

---

12. **Rick Regan**

    January 16, 2015 at 3:32 pm

    @swat,

    Please give me more context — what are you looking for?

---

13. **swat**

    January 17, 2015 at 5:06 am

    i m solving past papers of digital system design of our university,question mention as
    Q#3:a)how binary multiplication can be done?
    b)what are the four categories of binary multiplication?.
    i solve part (a) but cannot solve part(b).
    plz solve it.thanks

---

14. **Rick Regan**

January 17, 2015 at 10:40 am

@swat,

I still don’t understand “categories” in the context of your course. Sorry.

15. **nadeem**

February 23, 2015 at 4:56 am

u have explained it very clearly thanks

16. **Hassan**

May 5, 2015 at 2:36 am

Just wanted to say thanks for such a wonderful explanation. This really helped me in my Algorithm Analysis exam. 🙂

17. **John**

May 14, 2015 at 7:12 pm

Hi Rick thanks for the assistance in converting fractional decimals to binary but what do I do when the fractionals just keep repeating as I multiply by 2? Is there a way to tidy it up?

18. **Rick Regan**

May 14, 2015 at 8:19 pm

@John,

You must have intended your comment for a different article (http://www.exploringbinary.com/base-conversion-in-php-using-bcmath/ ?). In any case, if it keeps repeating, just note that in the answer. For example, 0.1 decimal becomes $0.0\overline{0011}$ binary.

19. **John**

May 15, 2015 at 2:09 pm

Rick my question is in relation to your above explanation about converting fractionals to binary.

“Now do the fractional part:

0.1 * 2 = 0.2
0.2 * 2 = 0.4
0.4 * 2 = 0.8
0.8 * 2 = 1.6
0.6 * 2 = 1.2
0.2 * 2 = … (repeats)

The answer for the integer part is 0.00011001100110011…”

I was wondering, instead of constantly multiplying by 2 like in the example, if it could be safely rounded. It was more to do with other examples where the bits are more variable in their sequence but it was ongoing. If it could be rounded like you suggest in your response then that is perfect.

I am new to computer arithmetic so sorry if it seemed silly to ask.

---

20. **Rick Regan**

May 15, 2015 at 2:43 pm

@John,

Sorry, I didn’t know you were referencing the comments section; my earlier response was to another off-topic comment. (BTW, I just noticed a typo I made in that comment. I said ‘integer’ instead of ‘fractional’. I just fixed it.)

I don’t understand your question. $0.0\overline{0011}$ is the exact mathematical answer. In a computer, it would be rounded, depending on the format. For single-precision IEEE, it would be rounded to 24 significant bits: 0.000110011001100110011001101. You still need some algorithm to generate all those bits in the first place, so in that sense the "constantly multiplying by 2" is necessary.

---

21. **Akhil**

July 24, 2015 at 11:58 pm

If we perform binary multiplication for two numbers, and we will get some result. For some result, at adding place if we get four 1's, then should we take two carry bits to add for another number?

---

22. **Rick Regan**

July 25, 2015 at 10:01 pm

@Akhil,

Effectively, yes.

For decimal, the analog would be if a column added up to 100 or more. (Not that you would ever have a multiplication in practice with that many partial products.) Let's say the column sum is 215. You would write down a '5' and carry the '21' to the next column.

In binary you follow the same procedure. (This "composite carry" happens much more readily though — with as little as three partial products.) If your column adds to binary 101, for example, you write down '1' and carry the '10'.

This probably gets too confusing in binary, so it may be easier instead to do a series of additions, in pairs.

---

23. **Mitul**

August 16, 2015 at 1:43 am

101.11*111.01
I've problem in this binary operation.

provide me complete step by step solution with addition & carry generated during binary addition of this sum.....

---

24. **Rick Regan**

August 16, 2015 at 8:59 am

@Mitul,

Tell me more about what you did so I can see where you went wrong.

---

25. **vasundhara**

August 21, 2015 at 11:05 pm

(10111)2 * (1110)2 convert binary to hex decimal equivalent.

I need the procedure and answer for that is 142

---

26. **vasundhara**

August 21, 2015 at 11:30 pm

Rick Regan sir, can you please tel me the procedure.

---

27. **Rick Regan**

August 28, 2015 at 8:25 pm

@vasundhara,

The procedure is what this article is about. (BTW, the answer is 322 when converted to decimal.)

---

28. **SasQ**

January 15, 2016 at 10:25 am

It is worth mentioning that this algorithm of multiplication has been known over 2000 years ago by the Ancient Egyptians (at the very least). They used it for fast multiplication, though they weren't using binary digits all the way. Here's how their algorithm worked:

Let's say they needed to multiply 12×34.
They wrote powers of 2 in one column, and powers of the multiplicand (12) in another column, by repeating doubling, like this:
1 12
2 24
4 48
8 96
16 192
32 384
Each row is obtained by doubling the previous row.
Now they tried to compose the multiplier (34) from these powers of 2:
34 = 32 + 2
and they marked the rows corresponding to these powers of 2 with some mark, let's say a star:
1 12
* 2 24
4 48
8 96
16 192
* 32 384
To get the final result, it's enough to select the corresponding doublings of the multiplicand (12) from these rows which has been marked, and add them up:
24 + 384 = 408
which is the correct answer.
All these operations were very easy and natural for them, because they used an additive number system. So to double a number, one could simply double each symbol in the representation of a number (and collect them when needed into symbols of greater value, like with Roman numerals). To add them, they simply gathered all the symbols.

Why does this work?
Because both columns are pretty much geometric sequences base 2, they just use *different units* (or scale factors): the first column uses the "standard" unit, that is, "1", and the second column uses the "new" unit, that is, 12 (the multiplicand). So as we get the multiplier from summing up "native" powers of 2:
32×1 + 2×1 = 32 + 2 = 34

the same way we get the final result from summing up the powers of 2 expressed in the “new unit” (12):
32×12 + 2×12 = 384 + 24 = 408

Because that’s what multiplication really is, at the deepest level: a *change of units*. A property of a number is that it just takes the unit, whatever it is, and repeats it that particular number of times. So the number 34, when applied to the “old unit” (1), repeats it 34 times to get 34 (itself). But when applied to another unit (12 in this case), it repeats *that* unit instead 34 times, to get 34×12=408. The Egyptian algorithm simply breaks up this multiplication into several easier steps (doubling is easier than the full-blown multiplication) by expressing one of the numbers in binary, and then scaling *that* by the new unit.

How does it relate to binary multiplication?
Well, just represent all those doubled numbers from the other column in binary, and you will see that they are the same bit pattern, just being shifted by one bit to the left each row. So adding the rows which match the bits set in the multiplier 34 (or the powers of 2 it is made of), you add up only the selected shifted rows of the same number (the multiplicand, or the new unit).

29. **Jason**

May 16, 2016 at 11:57 am

Hello Rick.

Please explain to me how the binary point of the fractions is determined. I understand the process, but idk how you determined where to place the binary point. Thank you.

Jason.

30. **Rick Regan**

May 16, 2016 at 12:14 pm

@Jason,

Just like in decimal multiplication, just add the number of places in the multiplicand and multiplier. In the example, the multiplicand has two places, and the multiplier has one, so the answer has three.

---

31. **Kush**

July 14, 2016 at 10:10 pm

Rick I have a doubt about the partial product's part . Can u please briefly explain it? Thank You!

---

32. **Rick Regan**

July 15, 2016 at 9:03 am

@Kush,

What is your specific question? If it is about how to add them, then check out comments 3 and 4. If it is about how to place the point, check out comments 30 and 31. (If it is about something else, or you need more clarification, please let me know.)

---

33. **sadman**

August 26, 2016 at 5:43 am

what will be the way of doing (11011111*11111)

Comments are closed.