2 Base Arithmetic

Base 10 is the most common system used to carry out routine arithmetic operations such as adding, multiplying, subtracting, and dividing. Simple calculators, for example, are built to automatically display numbers in base 10 despite the fact that calculators store the numbers in memory in binary. More sophisticated calculators contain special keys that will perform these arithmetic operations in other bases as well. It is also possible to build electronic circuits that can add and subtract in binary.

In order to truly understand low-level programming, however, it is very useful to be able to perform the following arithmetic operations by hand:

- adding in binary,
- multiplying in binary,
- subtracting in binary, and
- dividing in binary.

These operations behave in a similar fashion to their base 10 counterparts because they all rely on place value. In the decimal number system, for example, numbers have place value. The first column (from the right) is the units column, the next the tens column, the next the hundreds, and so on. Each successive column is multiplied by ten, since the counting system being used is base 10.

Binary numbers also have place value. In binary, the first column is the units column, the next the 2s column, next the 4s column (2 squared), the next the 8s column (2 cubed) and so on. Each successive column is multiplied by 2 since the counting system being used is base 2.

Roman numerals on the other hand, do not have place value and therefore do not behave well under the four fundamental arithmetic operations. The decimal number 345 is bigger than 99 simply because it has more digits (even though the leading digit of 3 is smaller than the leading digit 9). With Roman numerals, however, X (10 in base 10) is bigger than VIII (8 in base 10) even though X has fewer digits than VIII.

2.1 Adding in Binary

Adding in binary (base 2) is very similar to adding in decimal (base 10), where each column being added has a carry out to the next column. Here is a simple addition in base 10.

Example 1:

Add the base 10 numbers 346 and 578.

```
  346
+578
  924
```

In the right-most column, 8 plus 6 is 14. The 4 is written underneath and a 1 is carried to the next column and the addition is continued.

This same idea of carrying to the next column is used in binary. Here is an example.

Example 2:

Add the binary numbers 1101 and 1001.

```
 1101
+1001
10110
```

Therefore 1101 + 1001 equals 10110.

In the right-most column, 1 + 1 = 10. Remember that the counting is now in binary and not base 10. The 0 is written underneath and the 1 is carried to the next column to the left. In this column, the digits to be added are 0 + 0 + the carry of 1. The answer is 1. Therefore 1 is written underneath and a 0 is carried to the next column.

In the third column, the additions to be made are 1 + 0 + the carry of 0. The answer again is 1.

In the forth column the addition is 1 + 1 + a carry of 0. The answer is 10 (remember the addition is in binary) and therefore 0 is written underneath and a carry of 1 is written beside it since there are no more digits to be added.

This answer of 10110 can be checked by converting all the numbers to base 10 and then adding in base 10. (The method used to do this conversion is described in Section 4.1.3.) The binary number 1101 equals 13 in base 10. The binary number 1001 equals 9 in base 10. Finally, 10110 in binary equals 22 in base 10. Adding the base 10 numbers 13 and 9 equals 22 which is the same as our binary answer of 10110.

2.2 Multiplying in Binary

Multiplying in binary (base 2) is very similar to multiplying in decimal (base 10). Aligning the columns helps to avoid confusion when dealing with so many 1s and 0s.

Here are two examples of multiplying in binary.

Example 1:

In binary, multiply 10001 by 11110.

```
    10001
   x11110
    00000
   10001
  10001
 10001
10001
---------
111111110
```

Therefore $10001_2 \times 11110_2 = 111111110_2$. When the binary numbers are converted to base 10, it can be verified that 17 x 30 = 510.

Example 2:

In binary, multiply 1110 by 1011.

```
  1110
 x1011
  1110
 1110
0000
1110
10011010
```

Therefore $1110_2 \times 1011_2 = 10011010_2$

2.3 How People Subtract in Binary

Subtracting in binary (base 2) can be carried out using two very different methods. The first method, which is how people usually subtract in binary, is similar to subtracting in decimal (base 10) where each column being subtracted has a borrow from the next column.

Here are two examples of the first method of subtracting in binary.

Example 1:

In binary, subtract 1101 from 11101.

```
 11101
-01101
 10000
```

Therefore $11101_2 - 1101_2 = 10000_2$

Example 2:

In binary, subtract 111101 from 1110001.

```
 1110001
-0111101
  110100
```

Therefore $1110001_2 - 111101_2 = 110100_2$

2.4 How Computers Subtract in Binary

Unlike people, who subtract in binary (base 2) by subtracting with borrowing, computers subtract in binary by **complementing**. Complementing uses adding rather than subtracting. This method is used extensively by computers to subtract and also to store negative numbers. The method of subtracting using complementing also works in base 10 (in fact any base system). In base 10, the method works by, in effect, borrowing 99 in one step and paying it back in another step.

Here is an example of subtracting using complementing in base 10. Note that it uses addition rather than subtraction.

Example 1:

Subtract 43 from 52 using complements. To subtract using complements follow these steps.

1. Align the two numbers underneath each other making sure to include leading digits.

```
 52
-43
```

2. Leave the top number alone and take the complement of the bottom number. Since we are avoiding traditional subtracting, take the 43 and find what needs to be added to 43 to get 99. The answer is 56 since 43 + 56 = 99. We are in a 2 digit base 10 number so 1 is subtracted resulting in 99. Therefore 56 is the complement. By writing the top number down and complementing the bottom number, the question becomes

```
 52
+56
```

3. Add the two numbers to get 108.
4. Drop the leading digit so that 108 become 08.
5. Add 1 to the difference so that 8 becomes 9, as before.

Dropping the leading 1 does not seem logical at first but eliminating the leading 1 is equivalent to subtracting 100 from the sum. Subtracting 100 compensates for adding the 99 at the first. The 1 is added at the end to get back to 100.

Complementing seems unnecessarily complicated at first but this five-step method eliminates an entire arithmetic operation (subtracting) and also eliminates the need for the processor to have separate circuitry to process subtraction.

The next example also uses complementing to subtract in binary. This example, however, uses a method called **one's complement**. This method requires no subtracting. It uses only the operation of addition.

Example 2:

In binary, subtract 1101 from 11101. To subtract using one's complement follow these steps.

1. Align the two numbers underneath each other making sure to include leading digits.

```
 11101
-01101
```

2. Leave the top number alone and take the complement of the bottom number. Taking the complement in binary means interchanging any 0 with a 1 and any 1 with a 0. By writing the top number down and complementing the bottom number, the question becomes

```
 11101
+10010
```

3. Add the two numbers to get 101111.
4. Drop the leading digit so that 101111 becomes 01111.
5. Add 1 to the difference so that 01111 becomes 10000. Therefore $11101_2 - 1101_2 = 10000_2$.

These calculations can be checked by converting the binary numbers to base 10 and subtracting. Converting 11101 to base 10 results in the number 29. Similarly, 1101 results in 13 in base 10. Subtracting 13 from 29 gives the result 16. The base 10 number 16 is equal to the answer from Step 5, namely 10000.

Here is another example which uses one's complement to subtract in binary. This example is somewhat different because the first digit in the number being subtracted is a 1, which makes the addition more complicated.

Example 3:

In binary, subtract 1011101 from 1110101. To subtract using one's complement follow these steps.

1. Align the two numbers.

```
 1110101
-1011101
```

2. Leave the top number alone and take the complement of the bottom number. Taking the complement in binary means interchanging any 0 with a 1 and any 1 with a zero. By writing the top number down and complementing the bottom number, the question becomes

```
 1110101
+0100010
```

3. Add the two numbers to get 10010111.
4. Drop the leading digit so that 10010111 becomes 0010111.
5. Add 1 to the difference so that 0010111 becomes 11000. Therefore $1110101_2 - 1011101_2 = 11000_2$.