

Module 6 – Arithmetic techniques

Overview

The study of arithmetic techniques is fundamental to understanding how microprocessors manipulate numbers. This can be seen in the following concept map.

Microcomputer operation (concept map)

Objectives

At the completion of this module you will be able to:

- demonstrate the arithmetic for binary addition and subtraction
- describe the complements system and its use in both decimal and binary arithmetic
- show how the complements system is used with positive and negative numbers; and

6.1 Binary arithmetic

Let's take a closer look at the binary system and the rules governing mathematical operations. From previous work we recall for addition, the rules are:

| | | | | | | | |
|---|---|---|---|---|---|---|--|
| | | 0 | + | 0 | = | 0 | |
| | | 0 | + | 1 | = | 1 | |
| | | 1 | + | 1 | = | 0 | plus a carry of 1 to the next highest column |
| 1 | + | 1 | + | 1 | = | 1 | Plus a carry of 1 to the next highest column |

Consider now an example showing binary addition.

6.1.1 Example 1, addition

Add the following numbers in both binary and decimal form: $13_{10} + 10_{10}$.

| Decimal | | Binary |
|--|---|---|
| 13 | → | 1101 |
| + 10 | → | 1010 |
| <hr style="width: 50px; margin: 5px 0;"/> 23 <hr style="width: 50px; margin: 5px 0;"/> | | <hr style="width: 50px; margin: 5px 0;"/> 10111 <hr style="width: 50px; margin: 5px 0;"/> |

← carry to next highest column

For subtraction the rules are:

| | | | | | |
|---|---|---|---|---|---|
| 0 | – | 0 | = | 0 | |
| 1 | – | 0 | = | 1 | |
| 1 | – | 1 | = | 0 | |
| 0 | – | 1 | = | 1 | With a borrow of 1 from the next highest column |

Consider an example showing binary subtraction.

6.1.2 Example 2, subtraction

Subtract the following numbers in both binary and decimal form: $14_{10} - 11_{10}$.

| Decimal | | Binary |
|---|---|--|
| 14 | → | 11110 |
| – 11 | → | 1011 |
| <hr style="width: 50px; margin: 5px 0;"/> 3 <hr style="width: 50px; margin: 5px 0;"/> | | <hr style="width: 50px; margin: 5px 0;"/> 0011 <hr style="width: 50px; margin: 5px 0;"/> |

← borrow from next column

6.2 Complements arithmetic

The complements system of arithmetic is used in most computer systems as it permits logic manipulation easily. The **true** complement of a number is obtained by subtracting **successively** each digit of the number from another number which is the (appropriate base – 1) and then adding a 1 to the least significant digit, (LSD).

1. The true complement of a **decimal** number is called the **ten's** complement.
2. The true complement of a **binary** number is called the **two's** complement.

6.2.1 Example 1, decimal complements

Find the true complement of 57_{10} .

Using the previous definition we have:

(appropriate base for decimal numbers – 1) = $10_{10} - 1 = 9_{10}$

$$\begin{array}{r}
 99 \\
 -57 \\
 \hline
 42 \\
 +1 \quad \text{add 1 to the LSD} \\
 \hline
 43
 \end{array}$$

Therefore 43_{10} is the true complement of 57_{10}

We note this result could be obtained directly by subtracting 57 from 100, however as you will see later it is more convenient to use this method when using digital logic to perform these functions.

6.2.2 Example 2, binary complements

Find the true complement of 10101_2 .

(appropriate base for binary numbers - 1) = $10_2 - 1 = 1_2$

$$\begin{array}{r}
 1111 \\
 -10101 \\
 \hline
 01010 \\
 +1 \quad \text{Note: this is the inverse of the number to be complemented} \\
 \hline
 01011 \quad \text{add 1 to the LSD}
 \end{array}$$

Therefore 01011_2 is the true complement of 10101_2

To simplify the above process, for binary numbers, the **true** complement is found by inverting the each binary digit and adding 1 to the LSD.

Inverting the digits we have, 01010
 Adding 1 to the LSD we have, 00001
 The true complement is: 01011

Note if we do not add the 1 to the LSD we obtain:

- the **nines** complement for decimal numbers

and

- the **ones** complement for binary numbers

6.3 Subtraction by addition using true complements

The most important use of complemented numbers is that they can be used to perform **subtraction by addition**. This technique is used in all computers as it is more convenient to use a logic adder than a subtractor.

6.3.1 Modular number systems

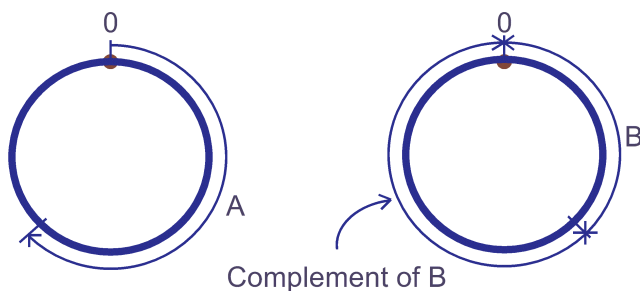
To perform subtraction by addition, using true complements, it is helpful to understand modular number systems. That is, a number system that consists of a finite number of elements. The modulus of a function was first introduced in module 5 where it was used to describe binary counters. The modulus of a counter was defined as the number of input pulses for one complete count cycle. Similarly, a modular number system counts to a maximum number then returns to zero, repeatedly.

For example, the odometer in a car may count to 99 999 kilometres then return to 00 000, in a similar way a clock counts to 12 then returns to 1. Both are modular number systems; the odometer having a modulus of 100 000; and the clock a modulus of 12.

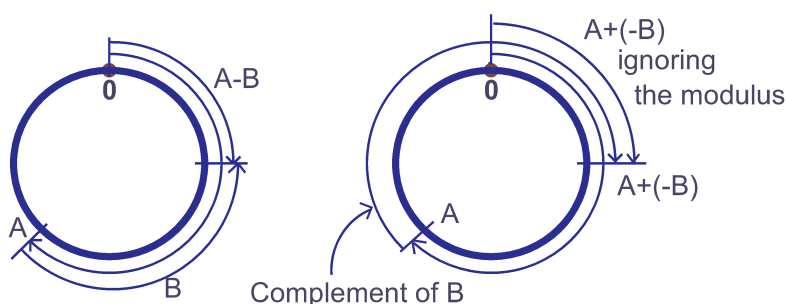
If, in the examples 6.2.1 and 6.2.2 you add the complement to the original number the result will be the modulus.

The subtraction by addition process can be illustrated graphically by assuming we have two numbers A and B on which the operation $A - B$ is to be performed.

Let the two numbers be represented by two arc segments of appropriate length around a circle. The circumference of the inner circle represents the modulus of the particular number system. The right-hand diagram shows that B plus the complement of B is equal to the modulus.



The next two diagrams depict $A - B$ using normal arithmetic on the left, and $A - B$ where the true complement of B is added to A, i.e. $A + (-B)$ on the right. To obtain the correct result the modulus (one complete circle) is ignored, so the outer arc represents the result of $A - B$.



The technique to perform subtraction by addition is now stated as three basic steps:

- The number to be subtracted (**subtrahend**) is complemented.
- The complemented subtrahend is added to the other number (**minuend**).
- If the modulus is now ignored then the result is a subtraction.

This technique is best illustrated by examples in both the decimal and binary systems.

6.3.2 Example 1, decimal complements arithmetic

Execute $60_{10} - 49_{10}$ using the true complements method.

Using the three basic steps:

- The subtrahend (49) is complemented thus:

$$\begin{aligned}(10 - 1) &= 9 \\ 99 - 49 &= 50 \\ 50 + 1 &= 51_{10}\end{aligned}$$

The modulus is $49 + 51 = 100$

- The complemented subtrahend (51) is added to the minuend. Thus:
 $51 + 60 = 111_{10}$
- The modulus (100) is now ignored and the result (11) is the required subtraction.

Try this using the circular representation. Note: once around the circumference is the modulus of this number system, i.e. 100, so 51 is a little over half way around the circle.

6.3.3 Example 2, binary complements arithmetic

Execute $10111_2 - 10001_2$ using the true complements method.

- True complement of 10001 is 01111_2 .
The modulus is $10001 + 01111 = 100000_2$.

- Add to minuend:

$$\begin{array}{r} 10111 \\ 01111 \\ \hline 100110_2 \end{array}$$

- Ignore the modulus and the result (110_2) is the required subtraction.

Try this using the circular representation. You may find it difficult to determine the arc lengths in binary, so convert each binary number to decimal as you proceed.

6.4 Subtraction using NINES and ONES complements

6.4.1 Example 1, nines complements

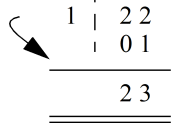
Execute $75 - 52$ using nines complements.

- Nines complement of 52 is 47.

- Add to minuend

$$\begin{array}{r}
 75 \\
 + \quad 47 \\
 \hline
 122 \\
 \hline
 23
 \end{array}$$

- This time add the carry bit to the result



23 is the required subtraction

6.4.2 Example 2, ones complements

Execute $11011 - 1010$ using ones complements.

Note: The subtrahend must have the same number of binary digits as the minuend prior to finding the complement, i.e. 1010 becomes 01010 prior to complementing.

- Ones complement of 01010 is 10101

- Add to minuend

$$\begin{array}{r}
 11011 \\
 + \quad 10101 \\
 \hline
 110000 \\
 \hline
 00001 \\
 \hline
 10001 \quad \text{Answer}
 \end{array}$$

Note: The extra digit which appears in the next highest column is added to the result and is called the ‘**end around carry**’.

The basic difference between using the tens or true complement and the nines or ones complement methods, is where and how the extra 1 is added.

It is easier for some computers to use the **ones** complement rather than **true** complements, so the ‘end around carry’ method is sometimes used.

6.5 Signed binary arithmetic

So far we have assumed that the subtrahend is smaller than the minuend which means that the result of a subtraction is always positive, but this is not always the case. For two numbers A and B we have four possibilities:

- Case 1** $A + B$ (both numbers positive)
Case 2 $A - B$ (B negative) $\implies A + (-B)$
Case 3 $-A + B$ (A negative) $\implies (-A) + B$
Case 4 $-A - B$ (both negative) $\implies (-A) + (-B)$

Again let us consider numerical examples to illustrate these situations.

For the two numbers A and B,

Let $A = +13_{10}$ and $B = +11_{10}$ in decimal

then $A = 00110_2$ and $B = 00101_2$ in binary.

Why add the leading zeros in binary?

- The binary number is increased to 5 bits to ensure the magnitude of the result in each case falls within the modulus of the number system.
 e.g. $13 + 11 = 24$, this result is too large to be represented by 4 bits, so increasing the number of bits by one gives us a range of 0 to 31_{10} , i.e. a modulus of 32_{10} .
- Another bit is added to represent the sign of the number, i.e. the MSB is the sign bit.
 - For positive numbers simply add a leading '0' as the MSB.
 - For negative numbers 2's complement the 5-bit number (magnitude bits) to make it negative, then add a leading '1' as the MSB.

| MSB | | | | | LSB |
|----------|------------------------------|---|---|---|-----|
| 5 | 4 | 3 | 2 | 1 | 0 |
| Sign bit | Magnitude bits | | | | |
| 0 = + | | | | | |
| 1 = - | | | | | |
| | 2's complemented if negative | | | | |

6.5.1 Case 1

A + B

Decimal

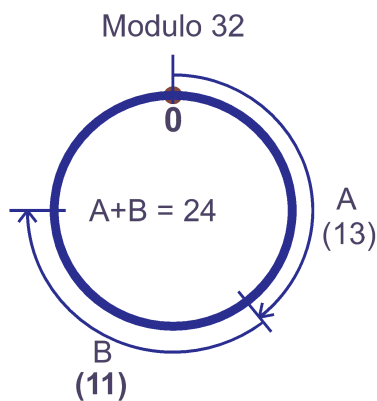
$$13 + 11 = 24$$

$$\begin{array}{r}
 001101 \\
 001011 \\
 \hline
 0011000
 \end{array}$$

← result
 carry bit sign bit

Note: Complements were not used in this case so there was no requirement for the number system to be modular. However, if the number system was modular the carry bit would be ignored; the sign bit indicates a positive result; and the magnitude is 11000 (24₁₀).

In circular representation:



6.5.2 Case 2

A - B

Decimal

$$13 - 11 = 2$$

Using true complements:

The true complement of B is $110100 + 000001 = 110101$

Add to minuend,

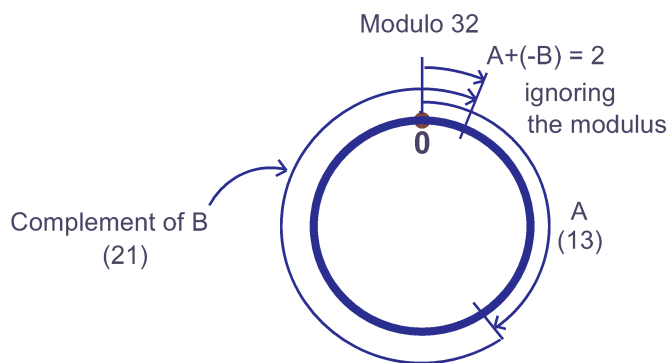
$$\begin{array}{r}
 001101 \\
 110101 \\
 \hline
 1000010
 \end{array}$$

← Result
 carry bit sign bit

Notes:

- The true complement of B is used in this case so the number system must be modular, and the modulus is 32_{10} .
- When using true complements we ignore the carry bit in the result and the remainder is the magnitude, i.e. 00010_2 (2_{10}).
- Since the sign bit is positive the magnitude is in the correct form.

In circular representation:



6.5.3 Case 3

-A + B

Decimal

| |
|-----------------|
| $-13 + 11 = -2$ |
|-----------------|

Using true complements:

The true complement of A is $110010 + 000001 = 110011$

Add to minuend,

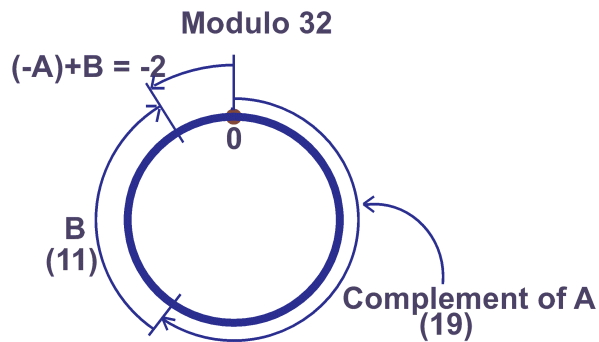
| | |
|-------------|----------|
| 0 0 1 0 1 1 | |
| 1 1 0 0 1 1 | |
| ----- | |
| 0 1 1 1 1 0 | ← result |

carry bit ↑ sign bit

Notes:

- The true complement of A is used in this case so the number system must be modular, and the modulus is again 32_{10} .
- When using true complements we ignore the carry bit in the result and the remainder is the answer, i.e. 11110_2 (30_{10}).
- However, the result is negative so the magnitude is in 2's complement form. Taking the 2's complement of 11110_2 , we get 00010_2 (2_{10}).

In circular representation:



6.5.4 Case 4

$-A - B$

Decimal

| |
|------------------|
| $-13 - 11 = -24$ |
|------------------|

Using true complements:

The true complement of A is $110010 + 000001 = 110011$

The true complement of B is $110100 + 000001 = 110101$

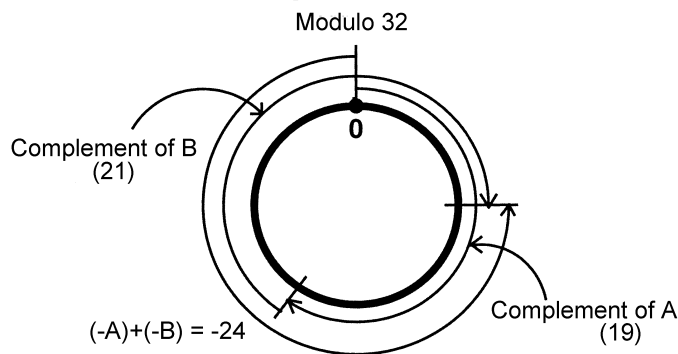
$$\begin{array}{r}
 110011 \\
 + 110101 \\
 \hline
 1101000
 \end{array}$$

← result
 carry bit sign bit

Notes:

- The true complements of A and B are used in this case so the number system must be modular, and the modulus is again 32_{10} .
- When using true complements we ignore the carry bit in the result and the remainder is the answer, i.e. 01000_2 (8_{10}).
- However, the result is negative so the magnitude is in 2's complement form. Taking the 2's complement of 01000_2 we get 11000_2 (24_{10}).

In circular representation:



Activity 6.1

1. Find the NINES complement and also the TENS complement of the decimal number 477.
2. Find the ONES complement and also the TWO's complement of the binary number 10001100.
3. Subtract the decimal number 21 from the decimal number 35 by converting both into binary format and using TWO's complement signed arithmetic.

Subtract the decimal number 25 from the decimal number 19 by converting both into binary format and using TWO's complement signed arithmetic.



Solutions to activity 6.1

1. The nines complement of 477 is 522, and the tens complement is 523.
2. The ones complement of 10001100 is 01110011, and the twos complement is 01110100.
3. Convert your answer back into decimal to check that it is correct.
4. Convert your answer back into decimal to check that it is correct.