International Journal of Advanced Science and Research
ISSN: 2455-4227; Impact Factor: RJIF 5.12
Received: 28-03-2020; Accepted: 15-04-2020; Published: 22-04-2020
www.allsciencejournal.com
Volume 5; Issue 2; 2020; Page No. 18-20

Research on improving the division operation of single-chip microcomputer by using shifting bisection scanning method

Hanyu Yang
College of physics and Electronic Engineering, Qilu Normal University, Jinan, China

Abstract
The single-chip microcomputer can control many aspects through a digital operation. In the division operation of a single-chip microcomputer, the method of subtracting while shifting (try to calculate) is generally adopted, so the time consumed by this operation cannot meet the requirements of fast operation. This is mainly due to the fact that the divisor needs to move to the left and enter the remainder area in turn, after the intermediate remainder is generated, it is compared with the divisor, which wastes a lot of time. To solve this problem, we can use the fast scanning method to improve, that is, to move in many bits each time, and then use the dichotomy to get the value of the quotient.

Keywords: SCM, division operation, scanning method

1. Introduction

The single-chip microcomputer (SCM) is also known as the microcontroller. It is a small and complete microcomputer system which uses VLSI technology to integrate CPU, Random-Access Memory (RAM), Read-Only Memory (ROM), I/O Port, Interrupt System, Timer/Counter and other elements on a silicon chip [1].

The SCM technology began in the 1990s, mainly used in the field of industrial control. It is developed from the special processor with the only CPU in the chip. The earliest design concept was to integrate a large number of peripheral devices and CPU into a single chip, making the computer system smaller and easier to integrate into the complex and strict volume requirements of the control equipment [1].

Nowadays, people's life has been inseparable from the SCM, whether it is the practice of automatic measurement or intelligent instrument, we can see the figure of SCM technology [2].

With the expansion of the application range of SCM, it needs to deal with more and more data and larger data. But, when the SCM carries out a large number of division operations, its processing speed will be reduced. This affects the working efficiency of the SCM. So, how to improve the speed of division operation of a large number of data by SCM? This paper analyzes and discusses this problem.

2. Problems and the operation principle of division of the SCM

The standard floating-point binary division is often used in the division operation of the SCM. It works like this: The standard floating-point binary division operation is divided into two parts: Order code operation and mantissa operation [3].The division of standard mantissa is realized by shift subtraction, that is to say, the dividend moves to the left enter the remainder area in turn, generates the intermediate remainder, and then compares the intermediate remainder with the divisor. If it is greater than the divisor, subtraction is performed, and the middle remainder becomes the result of the subtraction, and the quotient of the last place in the quotient area is 1; if the middle remainder is less than the divisor, the middle remainder remains unchanged, and the upper quotient of the last place in the quotient area is 0.Then, the quotient, the dividend and the remainder are shifted to the left by one bit until the dividend is finished [3-5].

For example: Calculate 150 divided by 3.

Lists the floating-point binary mantissa of these two numbers:

Decimal system: 150 3

Floating-point binary mantissa: 10010110 11

Table 1

Remainder region	Dividend	Quotient	Explanation
00000000	10010110	00000000	
00000001	0010110 0	00000000	The remainder region, dividend and quotient area are shifted one bit to the left.
00000001	0010110 0	00000000	Comparing the values of the remainder region and divisor. If remainder region is small, the value of the remainder region is unchanged, write ‘0’ to the last digit of the quotient.
00000010	010110 00	00000000	The remainder region, dividend and quotient area are shifted one bit to the left.
00000010	010110 00	00000000	Comparing the values of the remainder region and divisor. If remainder region is small, the value of the remainder region is unchanged, write ‘0’ to the last digit of the quotient.
00000100	10110 000	00000000	The remainder region, dividend and quotient area are shifted one bit to the left.
00000001	10110 000	00000001	Comparing the values of the remainder region and dividend. If dividend is small, the remainder region becomes the result of the remainder region minus divisor, write ‘1’ to the last digit of the quotient.

00000011	0110 0000	00000010	The remainder region, dividend and quotient area are shifted one bit to the left.
00000000	0110 0000	00000011	Comparing the values of the remainder region and dividend. If dividend is small, the remainder region becomes the result of the remainder region minus divisor, write '1' to the last digit of the quotient.
00000000	110 00000	00000110	The remainder region, dividend and quotient area are shifted one bit to the left.
00000000	110 00000	00000110	Comparing the values of the remainder region and divisor. If remainder region is small, the value of the remainder region is unchanged, write '0' to the last digit of the quotient.
00000001	10 000000	00001100	The remainder region, dividend and quotient area are shifted one bit to the left.
00000001	10 000000	00001100	Comparing the values of the remainder region and divisor. If remainder region is small, the value of the remainder region is unchanged, write '0' to the last digit of the quotient.
00000011	0 0000000	00011000	The remainder region, dividend and quotient area are shifted one bit to the left.
00000000	0 0000000	00011001	Comparing the values of the remainder region and dividend. If dividend is small, the remainder region becomes the result of the remainder region minus divisor, write '1' to the last digit of the quotient.
00000000	00000000	00110010	The remainder region, dividend and quotient area are shifted one bit to the left.
00000000	00000000	00110010	All dividends are moved to the remainder region. The operation is finished and the quotient area value is output.

3. The reason and analysis for the slow speed of division

When we use this method to manipulate the mantissa of two floating-point binary numbers, the dividend must be expanded to 2n-bit operands, and N subtractors are required. [6]

If the intermediate remainder is less than the divisor, you need to perform a shift addition. If the intermediate remainder is less than the divisor and the chance of being greater than the divisor is equal, for the standard mantissa division, it carries out 2n times of n-bit shift, n-degree numerical comparison, n-degree n-bit subtraction, and n-degree n-bit addition [7]. Therefore, if we improve the standard algorithm, we must try to shorten its shift time, addition, and subtraction time.

4. The method of solution

We can use the fast-scan method to improve it. When moving bits, the fast scanning not only forms one quotient but also forms multiple quotients in once shift, which can reduce the shift period and improve the execution speed [8].

The division is the inverse operation of multiplication, so we can use the reverse reasoning of the multiplication scanning method to achieve division.

First, compare the values of the divisor and dividend. If the divisor is small, write '1' to the last digit of the quotient. Then the divisor length is calculated. If the divisor is less than 4 bits, the divisor and quotient area is shifted to the left by 4 bit and output, if the divisor is greater than 4 bits, the divisor and quotient area will be moved to the left by the same length as the divisor, and then the divisor and quotient area will be output. When the dividend is less than the divisor, the divisor length is calculated directly, if the divisor is less than 4 bits, the divisor and quotient area is shifted to the left by 4 bit and output, if the divisor is greater than 4 bits, the divisor and quotient area will be moved to the left by the same length as the divisor, and then the divisor and quotient area will be output.

Now, expand the remainder area compared with the divisor to $n+4$ digits including the high byte of the dividend output. Because after output, the dividend is less than the divisor, so it is shifted to the left by 4 bits is equivalent to a 16-fold expansion. So, suppose now the multiple of the divisor in the remainder area is T, then $0 \leq T < 16$, Because, if T is greater than or equal to 16, prove that the remainder area before the shift operation is greater than the divisor. In summary, this is impossible. In this way, when the value of T is determined, a 4-bit quotient value Y can be generated in one shift operation. $Y=[Yn3, Yn2, Yn1, Yn0]_4$, $(0 \leq Y < 16)$.

Use dichotomy to determine the value of T: Because Y=[Yn3, Yn2, Yn1, Yn0]4, there are 16 possible values for Y, so $0 \leq Y \leq 15$. Assuming the remainder value is A, use the shift method to find 2*A, 4*A, 8*A, K*A ($1 \leq K \leq 15$),and store the result in the cache area, because the remainder is also in the cache area, first compare the remainder with the value of 8*A, if the remainder is greater than the value of 8*A, Then compare with the value of 12*A again; if the remainder is less than the value of 8*A, compare the remainder with the value of 4*A. Since $2^4 = 16$, in this calculation mode, the value of Y can be obtained by performing up to 4 calculations.

After the Y value is obtained, perform the above steps again until all the dividends are serialized, and then sort and output obtained Y values.

5. Conclusion

The SCM has a wide range of applications in various fields of contemporary science, technology, and life. It is mainly controlled by digital operations.

In the division operation of a SCM, the method of subtracting while shifting (try to calculate) is generally adopted, the time consumed by this operation cannot meet the requirements of fast operation. We believe that the main reasons for this problem are: The divisor needs to move to the left and enter the remainder area in turn, after the intermediate remainder is generated, it is compared with the divisor, which wastes a lot of time.

Therefore, we think that we can use the fast scan method to improve, that is, shift multiple bits at a time, and then use the dichotomy to find the value of the quotient. Since this method does not perform addition operations, the operation speed is accelerated. This method can scan the quotient value of at least 4 bits at a time, which greatly reduces the calculation time.

However, this method increases the program writing length due to the complicated judgment method. At the same time, the operations are all performed in the cache area, which takes up too much memory. Therefore, when performing operations with a small amount of data, it is sufficient to use standard floating-point binary division. This new method is more suitable for situations where the amount of data is large and the calculation speed is fast.

6. References

1. Li QL. Principle and application of single chip microcomputer. Beijing: Tsinghua University Press, 2014.
2. Xu ZG. Technology and application of single chip microcomputer. Beijing: Post & Telecom Press, 2009.
3. Zhou DM. Microcomputer hardware software and its application. Beijing: Tsinghua University Press, 1988.
4. Chen XG. Principle and Application of Microcomputer. Beijing: Publishing House of Electronics Industry, 2012.
5. Zhao W. Wang HZ. Chen XL. Implementing Rapid Scan Floating Point Division Arithmetic of Multiplex Byted with a Single Chip Microcomputer. Journal of Jilin Institute of Chemical Technology, 2000; (21):8-9.
6. Hong W. Principle and Application of Microcomputer. Wuhan: Huazhong University of Science & Technology Press, 2007.
7. Qin ZQ. Application of C51 SCM and C Language Programming. Beijing: Publishing House of Electronics Industry, 2009.
8. Chen SL. Tian H. Rapid floating-point division algorithm and its implementation on MCS-51 series single chip computer. Journal of Shaanxi Normal University, 2004; (32):43-45.