MSP430 Family

October 23, 2016, Bulat Valeev

Lection 6. Calculations.

Challenges

Introduction

Float point standard

Float point operations

Taylor series

Examples

Task in the class

Result

Hometask

Challenges

What you should know at the end of the day.

- Learn how make complex calculations in the MCU without float point unit
- Use approximation algorithms to make calculation faster
- Implement multiplication algorithms

Introduction

There are a lot of complex operations which is implemented in the embedded systems.
The multiplication is one of the most common operations in the MCU device.
In comparison with the summation, multiplication is much more complex and takes more iterations.

Multiplication

The simplest way to multiply two numbers is summation with shift.
We can present multiplication as summation of first operand shifted n times if the n-th element of second operand equal to one.
The complexity of the task become N summations in the worst case.
Where N is number of the bits in number.

Low power MCU

The MCU's have the hardware multiplication usually, but low power MCU almost always doesn't have it, because it is too expensive. According to that fact, developer must implement multiplication by himself, of find open source library.

Calculation ways

Why multiplication is so important?
The fact is that the almost all mathematical operations as the sin, cos, e^x and others are approximated as number of multiplications and you will use the multiplication everywhere.

Float point standard

There is one main standard of the float numbers presentation: IEEE-754 single precision

1 bit for sign

8 bits for exponent.

23 bits for mantissa.

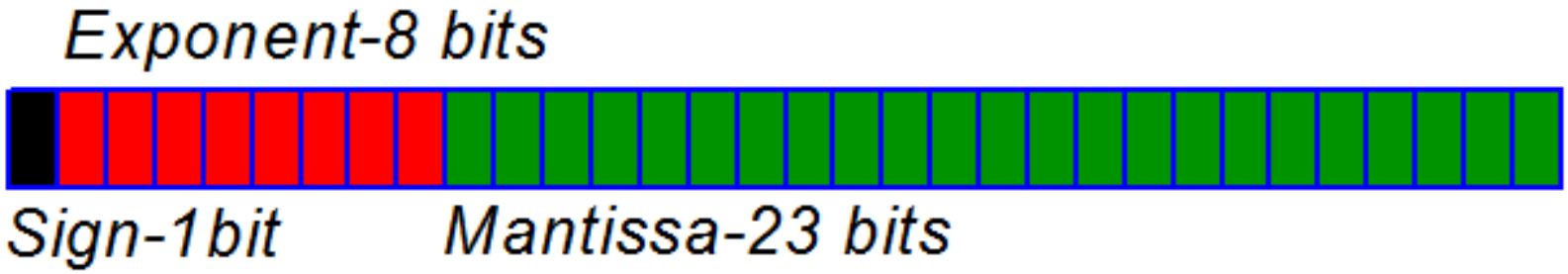

Figure: *Figure of the float point value*

Float point standard

IEEE-754 single precision covers negative and positive values from -127_{10} to 127_{10} or 11111111 exponent. In the mantissa value assumed that first (24-th bit) equal to 1. This notation called "normalized" and allows to use unique representation of the numbers.
Example:sign=1. Exponent=10001010.
Mantissa=(1)00001000011110000000000 Solution: Sign negative.
Exponent=$2 + 8 + 128 - 127$=11.
Mantissa=$1_{0+11} + 1_{-5+11} + 1_{-10+11} + 1_{-11+11} + 1_{-12+11} + 1_{-13+11} =$
$2048 + 64 + 2 + 1 + 0.5 + 0.25$=2115.75 Result=$-2115.75$

Float point operations

The multiplication and summation of the float point number become quite simple.
Summation:

- Normalization of the Exponent values with Mantissa shift.
- Addition of substitution of the Mantissas depending of the sign
- Normalization of the result in Mantissa and exponent

Multiplication implementation

The multiplication of the float point is the following:

Sign addition without overflow.

Exponent conversion to the scientific value $(-1, -2, 10.etc)$

Exponent addition.

Multiplication of the Mantissas with respect to the first (1.)

Mantissa and Exponent normalization

Taylor series

All mathematical functions possible to approximate as sum of the simpler functions. The way to approximate function is the Taylor series calculation.
Taylor series allows to represent hard function as the number of simple operations. The Taylor series are based on the derivatives which leads to the differential math implementation in the MCU.

Taylor series

The main equation of the approximation is presented in this form with unknown additional part.

$$f(x) = \sum_{k=0}^{\infty} \frac{f^{(k)}(a)}{k!}(x-a)^k \tag{1}$$

When we calculate approximate version of the function we take only few of sums instead of the all infinite series. The precision and the computational complexity depends from each other.

Examples

Many special functions have predefined simple way to calculate them.

$$sin(x) = x - \frac{x^3}{3!} + \frac{x^5}{5!} + ..$$

$$cos(x) = x - \frac{x^2}{2!} + \frac{x^4}{4!} + ...$$

$$e^x = 1 + \frac{x}{1} + \frac{x^2}{2!} + ...$$

Task in the class

Calculate multiplication of the two numbers: -120 and 10, with conversion in the float number.

Result

Result

Hometask

Implement simple float multiplication in the code without special NaN and infinite consideration.

Thanks for your attention