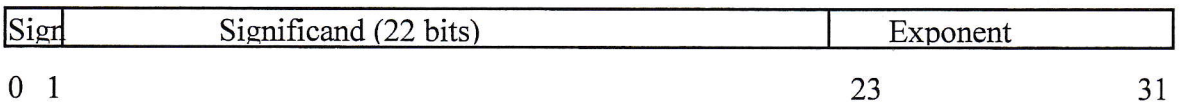
Homework Assignment 5 – ARM Assembly Programs (*Due date: March 11th)*

In networking programming there is software called middleware, hidden from the user. Some part of the middleware is so called ***marshierer***, responsible to transfer and convert the information from one computer to another. For example, from big endian to little endian byte order, from ASCII to EBCDIC code, from one data format to another and so on. Difficult part of a ***marshierer***, is to convert the floating-point data from one format to another with minimal loss of information. You are to implement such conversion program from Tandem floating point format to the IEEE 754 format.

IEEE 754 format. Single floating-point format has 1 sign bit, 8 bits exponent encoded in Excess 127 code, 23 bits significand in binary normalized form with an implicit integer bit.

Tandem Non-Stop Series (TNS) are super reliable and widely used supercomputers, which are used as servers in banks, financial institutions, military, etc. The TNS word (two bytes) is placed in memory at a two-byte memory boundary.

A single precision floating-point number is presented by a TNS doubleword. A four-byte word in memory is always fully aligned on a four-byte memory boundary.



The single precision floating-point number consists of one bit sign, 22-bit significand (fraction) and 9 bits exponent. The floating-point arithmetic uses the TNS number format, which predates and is different from the IEEE standard 754, which you know very well.

The fraction of a TNS-format floating-point number is always normalized, to be greater to 1 and less than 2. The high order integer bit is therefore dropped and assumed to have the value of 1 (hidden integer part 1.xx). During all calculations, the sign is temporarily removed and the assumed integer bit reinserted. The integer plus the 22 fraction bits are equivalent to 6.9 decimal digits; the 55 bits of an extended number are equivalent to 16.5 decimal digits. If the value of the number to be represented is zero, the sign is zero, the fraction is 0, and the exponent is zero.

The significand (fraction) of a floating-point number is a binary number with the binary point always between the assumed integer bit and the high-order fraction bit. The exponent part of the number, bits 7 through 15 of the low order word indicates the power of 2 multiplied by 1 plus the fraction. This field can contain values from 0 through 511. To express numbers of both large and small absolute magnitude, the exponent is expressed as an *excess-256* value; that is, 256 is added to the actual exponent of the number before it is stored. The exponent range is therefore actually -256 through +255.

The sign is explicitly given in the high-order bit; a 0 is positive, and a 1 is negative. Unlike integers, which use two's compliment, negating a floating-point number affects the only high-order bit and not the fraction or exponent fields. This is a typical sigh-magnitude code.

The absolute-value range of 32-bit floating-point numbers is $\pm 2^{-256}$ through $\pm(1- 2^{-23})* 2^{256}$. In decimal notation, this is approximately $\pm 8.64 * 10^{-78}$ through $\pm 1.15 * 10^{77}$.

You are to:

1. Define some real number (for example, 31.75_{10}) in TNS single precision floating-point format (Big Endian). Define the same number in IEEE 754 single FP format. Declare the numbers using DCD.

2. Write and debug an assembly subroutine, which converts TNS format in IEEE 754 single floating-point format.

3. Write and debug an assembly subroutine, which converts IEEE 754 single floating-point format to TNS format.

3. Write a main subroutine, which calls the conversion subroutines and compares the source number with the results (should be equal).

Remark: Please present:

- (MUST) the source code and the project of your program (sent by e-mail).
- Printout of the programs and results.