System Programming

DHBW Stuttgart, 2017
Prof. Dr.-Ing. Alfred Strey

Project assignment TINF15A

The MIPS floating-point coprocessor only supports all four basic arithmetic operations for both single and double precision IEEE 754 numbers. To achieve a very high performance for many scientific and technical applications, it might be useful to implement also the calculation of the trigonometric functions at machine level.

1) The first task consists in writing a function to implement sin(x) in MIPS assembly language. Here a Taylor series can be used for the approximation:

$$\sin(x) = \sum_{i=0}^{\infty} (-1)^i \frac{x^{2i+1}}{(2i+1)!} = x - \frac{x^3}{3!} + \frac{x^5}{5!} - \frac{x^7}{7!} + \cdots$$

The following steps have to be performed:

- First implement a high-level program (preferably in C) that approximates sin(x) according to the formula above and compares the result with the corresponding library function. Use the IEEE double precision format here! How many terms of the Taylor series are required for a good approximation, especially if x is close to 0?
- Now implement in MIPS assembly language a function `double sin0(double x)` that calculates sin(x) only for an argument x in the range $-\pi/2 \le x \le \pi/2$.
- Due to the symmetry and periodicity of the sin(x) function the calculation can easily be extended to support arbitrary positive and negative arguments. So implement a general function `double sin(double x)` that first reduces the argument to a number in the range $-\pi/2 \le x \le \pi/2$ and then calls internally the function `sin0`.
- Test and debug your MIPS functions `sin0(x)` and `sin(x)` with SPIM!

2) The second task consists in implementing the two other trigonometric functions cos(x) and tan(x) in MIPS assembler language.

- Write a function `double cos(double x)` in MIPS assembly language that calculates the cosine by calling internally the function `sin0(x)` or `sin(x)`.

  Remark: $\cos(x) = \sin(\pi/2 - x)$
- How can the trigonometric function tan(x) be realized based on the already encoded functions? Implement the MIPS function `double tan(double x)` for arbitrary values of x.
- Test and debug your implementations of cos(x) and tan(x) with SPIM.

3) Finally write in MIPS Assembler a test program `main()`, that asks the user for three values n, x_{min}, x_{max} and then calculates and prints all values $\sin(x_i)$, $\cos(x_i)$ and $\tan(x_i)$ for n equidistant values x_i in the interval $[x_{min}, x_{max}]$ as a formatted table on the screen.

Test your main program with SPIM!

4) **Optional** improvements:

- Optimize all MIPS functions code for highest efficiency!
- Try to determine an optimal number of terms for the Taylor approximation dependent on the argument x.
- $\tan(x)$ is not defined for $x = k\pi + \pi/2$ with $k \in Z$! Check for this special case and implement a suitable special treatment.
- Specify the worst case accuracy of your approximation.

General remarks:

1. Respect the requested calling hierarchy. Use the stack for saving registers. It is not necessary here to use the frame pointer `$fp`!
2. Stick to all MIPS conventions!
3. Comment each line of your assembler source code if useful!
4. The usage of MIPS pseudo instructions is allowed and strongly recommended!
5. Teams of up to two students are admitted.
6. Please submit your project files by email to strey@lehre.dhbw-stuttgart.de before the agreed deadline. The submission (one for each team) should contain the high-level source files, the commented assembler sources (runnable on SPIM simulator) and a short project documentation (max. two A4 pages) as pdf file.