# HW 6 - ARM floating-point and computing $e^x$

Due Monday March 29th by 5PM.

Euler's number is a mathematical constant ***e = 2.71828...***, and, like $\pi$ , is an irrational number (i.e., goes on forever without repeating). ***e*** arises in nature in various contexts, such as exponential growth and decay. The function $e^x$ is ***e*** raised to the power ***x***, where ***x*** is a real number (double). Most programming languages provide a function **`exp(x)`** for computing $e^x$**.** There is a nice formula for approximating $e^x$

$$e^x = \sum_{k=0}^{\infty} \frac{x^k}{k!} = \frac{x^0}{0!} + \frac{x^1}{1!} + \frac{x^2}{2!} + \frac{x^3}{3!} + \cdots + \frac{x^k}{k!} + \cdots$$

Calculus students might recognize this as the Taylor series expansion of $e^x$. The notation ***k!*** Is *k-factorial* and is ***k·(k-1)·(k-2) … 2·1***. By definition ***0! = 1***. In this homework you are going to write a C and ARM assembly language program to compute $e^x$. Do this by writing a function **`myexp(x)`** that computes and returns $e^x$ (we are naming the function **`myexp`** so it doesn't clash with the C math library function **`exp`**). Here, ***x*** can be a floating-point number and should be passed as a parameter. Obvioudly do not use the builtin **`exp`** function to implement **`myexp`**. Use the formula above. You cannot compute this for an infinite number of terms, so experiment and figure out how many terms you need by comparing your output to the result of the math library **`exp(x)`** function. Factorials grow very large quickly so the terms vanish to zero quickly as well.

Create a folder **`hw6`** and put in four files **`exp.c`** and **`exp.s`** (and **`exp.h`**) that implement **`myexp(x)`** in both C and assembly. Create a **`main.c`** program that takes a command line argument and outputs **`myexp(x)`** rounded to six decimal places.

```
./exp 1
2.718218

./exp 2
7.389056

./exp -5.2
0.005516
```