Enel452 – Fall 2024 – Assign 4 LUT

Given: before [2024-10-31 Thu]
Due: [2024-11-07 Thu] 23h55

1. (**10pts**) It is quite common in embedded systems to replace complex calculation with lookup tables, speeding up the calculation at the expense of using more memory.

   Your task is to code a lookup table that will speed up calculation of $\sin(x)$, avoiding all use of floating point arithmetic. Your table should work for any angle in the range -359.5 to $+$359.5 degrees, with a 0.5 degree resolution, however you should scale and translate the angle, so the actual "angle" will be an integer index into the table. The returned value from the table should also be an integer representing the sin value scaled by 100,000. Either bounds-check the input integer or describe the bounds (as in Design by Contract) in the function's header comment. Again, use table increments of $\Delta x = 0.5$ degrees.

Timing

No attempt at code speed-up should be performed without *measuring* to ensure you got a speed-up. Hence you must measure the speed of your LUT, and compare it to the speed of the built-in standard C `sin()` function.

Use an appropriate timing technique like the standard C `clock()` function, or a built-in high-res timer, or a logic analyzer like the ADALM 2000. Note the Intel RDTSC instruction is no longer recommended for timing, unless one is willing to make great efforts to constrain the execution environment. Also it's specific to Intel so, not present on the ARM.

Run your benchmark on one of the following targets:

- the cortex-m3 board, or
- any general computer, provided your code can easily be compiled and run on `snoopy`.

Accuracy

Your code should also determine the *accuracy* of the LUT. You can use the built-in `sin` from the standard math library for this. This is available even on the cortex-

m3 target, it just won't run very fast! I suggest you document your accuracy in parts-per-million (ppm), or similar.

Write-up

Write up your results in a readme file: describe (in a sentence) the target environment, note the compiler version and compile switches, the date on which the benchmark was run, show a table with the LUT and `sin()` speeds and the speed-up (or slow-down!) factor. For example:

```
Target: Thinkpad T420s (quad Intel Core i5-2520M @ 2.5GHz)
Date: 2018-10-03
Compiler: g++ 5.4.0
Switches: -O3
Measurement: using clock() with 1e9 loops.

      Function           Speed
      ---------------------------
      table lookup       0.748 ns
      Built in sin()     99.2 ns
      Table Speedup      132.6x
```

For more recent compilers, you might see a less dramatic speed-up or even *no* speed-up.

The solution should be submitted as usual via git.