Floating Point
Case studies

Arquitectura de Computadores

Departamento de Engenharia Informática

Instituto Superior de Engenharia do Porto

Luís Nogueira (lmn@isep.ipp.pt)

Intel IA32 floating-point arithmetic

Here we highlight an idiosyncrasy of these machines that can seriously affect the behavior of programs operating on floating-point numbers when compiled with *gcc*.

IA32 processors, like most other processors, have special memory elements called registers for holding floating-point values as they are being computed and used. The unusual feature of IA32 is that the floating-point registers use a special 80-bit extended-precision format to provide a greater range and precision than the normal 32-bit single-precision and 64-bit double-precision formats used for values held in memory. All single- and double-precision numbers are converted to this format as they are loaded from memory into floating-point registers. The arithmetic is always performed in extended precision. Numbers are converted from extended precision to single- or double-precision format as they are stored in memory.

This extension to 80 bits for all register data and then contraction to a smaller format for memory data has some undesirable consequences for programmers. It means that storing a number from a register to memory and then retrieving it back into the register can cause it to change, due to rounding, underflow, or overflow. This storing and retrieving is not always visible to the C programmer, leading to some very peculiar results.

More recent versions of Intel processors, including both IA32 and newer 64-bit machines, provide direct hardware support for single- and double-precision floating-point operations. The peculiarities of the historic IA32 approach will diminish in importance with new hardware and with compilers that generate code based on the newer floating-point instructions.

Ariane 5: the high cost of floating-point overflow

Converting large floating-point numbers to integers is a common source of programming errors. Such an error had disastrous consequences for the maiden voyage of the Ariane 5 rocket, on June 4, 1996. Just 37 seconds after liftoff, the rocket veered off its flight path, broke up, and exploded. Communication satellites valued at $500 million were on board the rocket.

A later investigation showed that the computer controlling the inertial navigation system had sent invalid data to the computer controlling the engine nozzles. Instead of sending flight control information, it had sent a diagnostic bit pattern indicating that an overflow had occurred during the conversion of a 64-bit floating-point number to a 16-bit signed integer.

The value that overflowed measured the horizontal velocity of the rocket, which could be more than 5 times higher than that achieved by the earlier Ariane 4 rocket. In the design of the Ariane 4 software, they had carefully analyzed the numeric values and determined that the horizontal velocity would never overflow a 16-bit number. Unfortunately, they simply reused this part of the software in the Ariane 5 without checking the assumptions on which it had been based.

Practice problem

The imprecision of floating-point arithmetic can have disastrous effects. On February 25, 1991, during the first Gulf War, an American Patriot Missile battery in Dharan, Saudi Arabia, failed to intercept an incoming Iraqi Scud missile. The Scud struck an American Army barracks and killed 28 soldiers. The U.S. General Accounting Office (GAO) conducted a detailed analysis of the failure [72] and de- termined that the underlying cause was an imprecision in a numeric calculation. In this exercise, you will reproduce part of the GAO's analysis.

The Patriot system contains an internal clock, implemented as a counter that is incremented every 0.1 seconds. To determine the time in seconds, the program would multiply the value of this counter by a 24-bit quantity that was a fractional binary approximation to 1. In particular, the binary representation 1 10 of 10 is the nonterminating sequence 0.000110011[0011] . . .2, where the portion in brackets is repeated indefinitely. The program approximated 0.1, as a value x, by considering just the first 23 bits of the sequence to the right of the binary point: x = 0.00011001100110011001100.

A. What is the binary representation of $0.1 - x$?

B. What is the approximate decimal value of $0.1 - x$?

C. The clock starts at 0 when the system is first powered up and keeps counting up from there. In this case, the system had been running for around 100 hours. What was the difference between the actual time and the time computed by the software?

D. The system predicts where an incoming missile will appear based on its velocity and the time of the last radar detection. Given that a Scud travels at around 2000 meters per second, how far off was its prediction?

Normally, a slight error in the absolute time reported by a clock reading would not affect a tracking computation. Instead, it should depend on the relative time between two successive readings. The problem was that the Patriot software had been upgraded to use a more accurate function for reading time, but not all of the function calls had been replaced by the new code. As a result, the tracking software used the accurate time for one reading and the inaccurate time for the other.