What a Fast FPU Means for Algorithms: A Story of Vector Elementary Functions

Marat Dukhan
mdukhan3@gatech.edu

Contribution

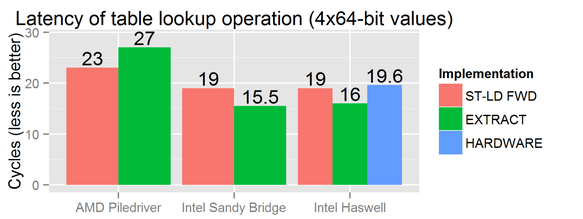
Over time, the balance between performance of FPU and table lookups shifted towards more FPU power. We bring two contributions which exploit this change.

1. We suggest three design rules for high-performance vector elementary functions (log, exp, cos): they should avoid table lookups, avoid division and square root operations in favor of multiplication and addition, and avoid branches in primary code path.

2. We evaluated the three design rules by implementing and benchmarking an experimental library of vector elementary functions, which demonstrates competitive performance and accuracy. The library is available on www.yeppp.info.

Rule 1: Avoid Table Lookups

Table lookups are hard to use with SIMD instructions. To perform a table lookup operation, first the indices are extracted from components of SIMD register to general-purpose registers (alternatively, the index register can be stored to memory and loaded into general-purpose registers element-by-element via store-load forwarding), then table lookups are performed and individual elements are loaded into different SIMD registers, and finally the loaded lanes of SIMD registers are combined with a shuffle instruction.

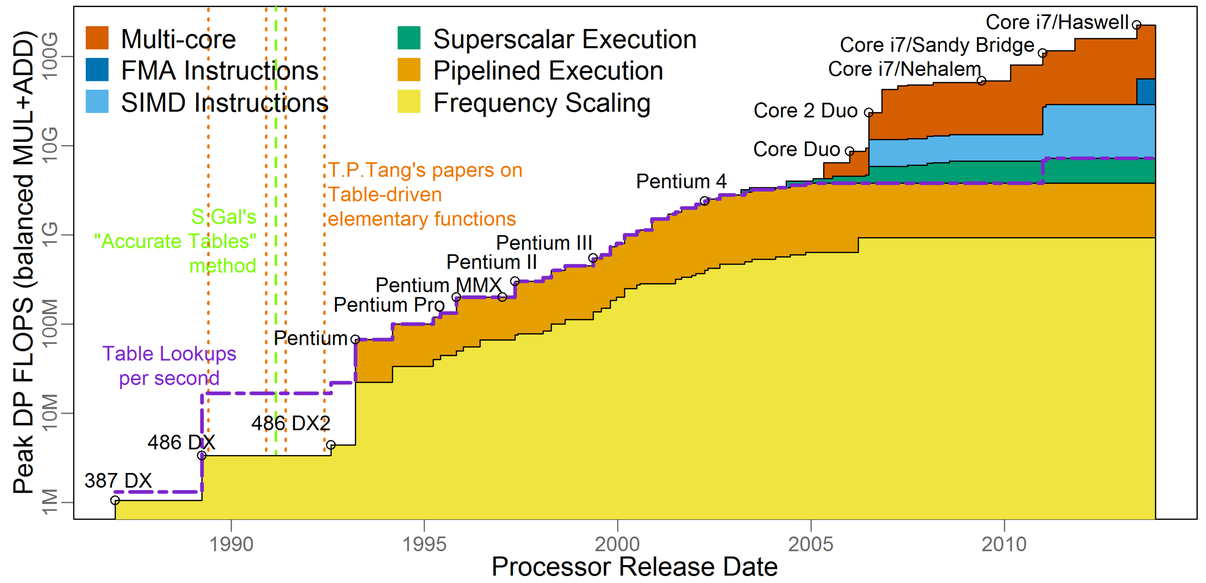


The total latency of such table lookup operations is too high to hide it with software pipelining or out-of-order execution. Although the recent Intel Haswell microarchitecture provides a hardware GATHER instruction, it does not reduce the latency.

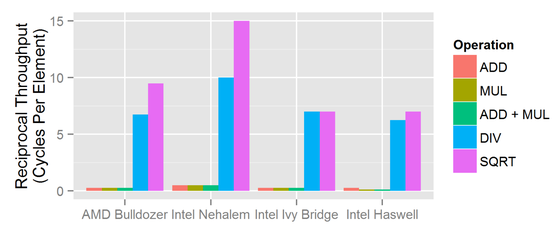
Conference Paper

Extended version will be presented on PPAM'13.

Evolution of Performance: FLOPS vs Table Lookups



Rule 2: Avoid DIV/SQRT



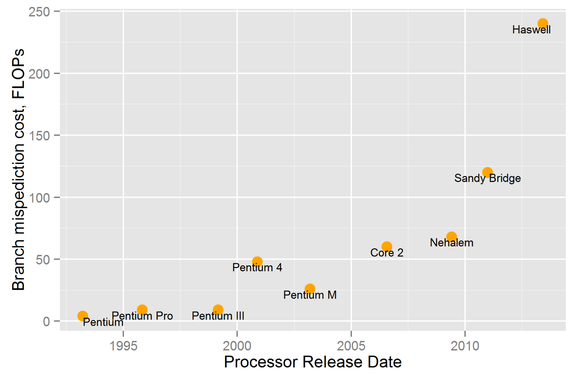
While most processors can issue multiplication and addition each cycle, division and square root are not pipelined, and have much lower throughput. We suggest that these operations should be avoided in favor of polynomial approximations and Newton-Raphson iterations.

Support

This work was supported by my advisor, Richard Vuduc, through The National Science Foundation under NSF CAREER award number 0953100, The U.S. Dept. of Energy, Office of Science, Advanced Scientific Computing Research under award DE-FC02-10ER26006/DE-SC0004915, and grants from the DARPA Computer Science Study Group program.

Rule 3: Avoid Branches

Piecewise approximations are widely used in elementary functions algorithms to reduce the order of polynomial approximation, but on modern processors the associated branch misprediction cost is prohibitively high, and the historical trend suggests that the situation will not improve in the future.



For this reason we avoid piecewise approximations and use branches only to detect special cases (such as NaN or infinite input), which do not normally happen and thus are predictable.

Background

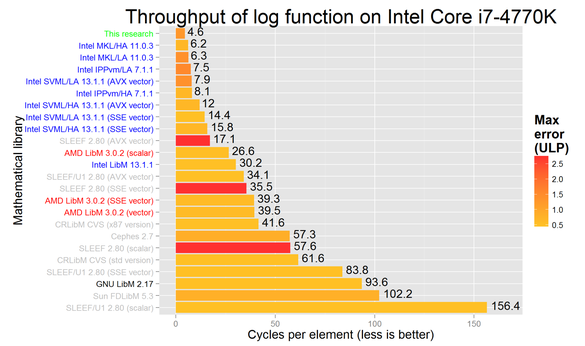
Vector elementary functions are mathematical functions, like log or exp, which independently operate on elements of a vector. In the early 1990th S. Gal and T.P.Tang suggested table-based algorithms with reliably high accuracy, and their variants dominated implementations of elementary functions since then. These algorithms use low-order polynomial/rational approximation of a function around tabulated pivot points, and combine two desirable properties:

- By smartly designing lookup tables they achieved good accuracy, and dense table values permitted low-degree approximations.

- Low-degree approximations only need a few FLOPs to evaluate, and result in good performance, especially on poor FPUs of early 90th.

But FLOPS numbers gradually outgrew table lookup metrics, making space for new algorithms.

Results: log/Intel Haswell



Results: exp/AMD Piledriver