SYLLABUS for CS 279 : as of Apr. 11, 1988

COMPUTER SYSTEM SUPPORT for SCIENTIFIC and ENGINEERING COMPUTATION

by Prof. W. Kahan
Elect. Eng. & Computer Science Dept.
University of California at Berkeley

This course, CS 279, is offered every two or three years by the Elect. Eng. & Computer Science Dept., drawing from the topics and readings listed below. Students who take the course earn credit by carrying out projects that, as often as not, add to the list.

(THIS IS A DRAFT. ITS READING LIST IS NOT YET COMPLETE.)

0. The environment for computing

Gresham's Law: "The Bad drives out the Good"; similarly, the Fast drives out the Slow even if the Fast is wrong.
Computer Users: Sophisticated but numerically naive.
Many layers of software from diverse sources.
Portable software in the public domain: LINPACK, EISPACK, ...
Commercial Libraries: IMSL, NAG, ... [Rice 1983]
Standards: for programming languages but little else.
Proprietary rights, Copyright, Patents, Trade secrets.
- L. J. Comrie's trick (see Preface to [Comrie 195?])
- Execute-only codes, Copy protection

Liability & Exculpation: Law of Torts [Prosser 196?]
- Reliability means conformity to reasonable expectations
- Marketing claims vs. Absolute and Negligent Liabilities
- Obligation to correct errors, Change vs. Compatibility

1. Comparisons of Floating-Point Arithmetics in Computers

Conventional floating-point arithmetics: [Sterbenz 1973]
- Radix, Wordsize, Range/Precision trade-off:
  - Why is Binary best? When is Decimal best?

Commercially significant machines:

IBM 7090/7094, '360, '370, PC's		UNIVAC 11xx
CDC 6600, Cyber 1xx, Cyber 205		Burroughs 6500
DEC PDP-11, VAX	CRAYs	Hewlett-Packard 3000

Calculators by Hewlett-Packard, Texas Instruments
IEEE standard 754:
Mainframes: ELXSI 6400, Hewlett-Packard "Spectrum"
Chips: Intel 80x87, Motorola 68881/2, AT&T WE 32106
Nat'l Semi. 32081, Weitek 1164/5, AMD 29027
Fairchild Clipper, Analog Devices 3212/22
BIT, Inmos T800 ?
... used by IBM PCs, Sun III, Apple Macintosh II, ...

Nonconventional representations of real numbers:
Logarithmic (Radix 1)
Matsui-Iri [1983], Hamada [1987]: variable exponent width
Clenshaw-Olver [1987] symmetric level-index arithmetic
- why are they all bad ideas? (Demmel [1987])

Kornerup-Matula [1987] floating-slash for rationals.

Interval Arithmetic [Moore 198?, Hertzberger & Alefeld 198?]
- what it is and what good it is, very briefly.

Multi-Precision schemes:
Conventional "Double" and "Extended" precisions
T. E. Hull's [1987] "Numerical TURING" language
R. Brent's [198?] MP package for Fortran
R. Fateman's [197?] "Bigfloats" in MACSYMA
Kulisch-Miranker [1981] paradigm, IBM ACRITH [1983] package
Kahan's indefinitely Extended IEEE 754/854

Implementation of Algebraic Operations { + - * / rem √ }:
Fast carry-propagation (fast prefix calculation, VLSI)
Floating-point add/subtract, shifters, normalizers
Multiplication: iterative, Booth re-encoding
Wallace tree of carry-save adders
Division: higher-radix [G. Taylor 198?], preconditioned
iterative using fast multiply, can be grubby (CRAY)
can be cleaned up, can be tested [Kahan 1987]
Why should division be atomic?
Why try to keep (div'n time)/(mult'n time) < 4 ?
Remainder; how to make it interruptible
Square root by hardware, by software, and without division!
Rounding in theory: role of Guard-Round-Sticky bits
- re-rounding without double-rounding [C. Lee 1988]
- carry-free roundings (chop, jam, ROM)
Rounding as practiced on commercially significant machines.
DPROD, or Multiply-Add primitives
Conversions: of precisions, float-integer, BCD-Binary
Exceptions: Traps/Faults, Flags, Defaults, Presubstitution
Invalid operation, Divide by zero, Over/Underflow, ...
- as practiced on commercially significant machines
- partial overflow on a CRAY
- partial underflow on CDC Cyber 1xx
- gradual underflow in IEEE 754/854
- NaNs, Indefinites and Reserved Operands
Range extension via over/underflow tallies [Kahan 1966]
Precision doubling, and further extensions [Pichat 1972]
Ultra-wide multiplication (Karatsuba, Cook, Schoenhage, ...)

2. Models of Floating-Point Arithmetic for Programmers:
Purpose of models: promote correctness proofs and portability.
Axioms: A. van Wijngaarden's [1966] 32 axioms
W. S. Brown's [1981] 20 axioms
- defects in non-categorical axioms - too loose
Kulisch's Ringoids, Semi-Morphism; "Optimality"
- defects in over-specified axioms - too slow
Descriptions: J. H. Wilkinson's [1959] crude inequalities
P. H. Sterbenz's [1973] parametrization
Environmental inquiries in Fortran, ADA, ...
Prescriptions: Computer manufacturer's manuals
IEEE standards 754/854

Tests for correct floating-point arithmetic:
  Test batteries: the IEEE 754 test-vectors tape
  Test patterns: N. Schryer's [1980], NAG's [1987]
  Inquiries and consistency tests: MACHAR [Cody 1980]
    PARANOIA [Kahan 198?]
  P-adic tests for correctly rounded square root [Kahan 196?]
    for correctly rounded division [Kahan 1987]

Identities that persist despite roundoff:
  Weak monotonicity
  Cancellation: $p-q$ is exact if $1/2 \leq p/q \leq 2$.
    When does integer $m = (m/n)*n$ rounded ?
    When does $x = (x+y) - y$ rounded ?
    When does $sqrt(x*x) = |x|$?
  Avoiding Drift: during conversion [Matula 196?]
    during iteration [Kahan 197?]
      - Knuth-Reiser [197?] proof
    Harry Diamond's [198?] theorem

Error-Analysis:
  Conventional "Forward" approach [Hotelling 193?]
  "Backward" approach [Turing, Givens, Bauer, Wilkinson, ...]
  Combined approaches; the "Web" of relationships.
  Attempts to automate error-analysis [Miller 198?, ACRITH]
  Misconceptions corrected:
    Rounding errors are NOT infinitesimal.
    Rounding errors are NOT random variates.
    Unstable computations are often free from Cancellation!
    Cancellation is often a GOOD thing;
      - equation solving, extrapolation, divided differences
    Simulation is easy, prediction difficult.
      - removable singularities
    Backward error-analysis is an explanation, not a licence.
      - accuracy is not transitive under composition
    Precision does not limit Accuracy. [Dekker, Pichat]
  Why "Double+ precision" is a good rule of thumb:
      - double zeros, normal equations, divided differences
  What "Unstable," "Ill-conditioned," "Ill-posed" mean;
      - how to help an ill-posed problem get well.
  Interval Arithmetic vs. Running Error-Analysis
  Examples:
    - When arbitrarily high precision might not help.
    - Triangle with given side-lengths.
    - Quadratic and Cubic equations; ... betrayed by books.
    - Polynomial equations generally.
    - Linear systems of equations, eigenproblems.
    - Numerical quadrature, infinite series.
    - Trajectory problems, Boundary value problems.

3. Comparison of computers' Floating-Point Register Architectures:

One accumulator: IBM 7094, DEC PDP 10, GE/Honeywell 635
Stack-top: B5500, H-P 3000
Small stack: H-P calculators, Inmos T800, Intel 80x87
- indefinitely long stack intended for the 8087 .

Effect upon procedures' argument passing, register saving
Extended width: internally in H-P calculators
Kulisch-Miranker's hidden "Super-Accumulator"
- in registers: IBM 7094, DEC PDP 10, GE/Honeywell 635
- in IEEE 754/854: Intel 8087, Motorola 68881, WE 32106

Orthogonal registers: IBM '370, CDC 6600, NS 32081, CRAY
DEC VAX, ELXSI 6400
Weitek, AMD, AD, ... chips

Vector registers: CRAY; simulated vectors in CDC Cyber 205.
- fast vector arithmetic without vector registers.
- local Data-Flow; CYDROME

Influences upon programming languages' expression evaluation:
Precision(s) available
Mixed-precision expressions:
- fundamentalist's Fortran for IBM '370, DEC VAX
- widest available for C on a PDP-11 ; i8087, m68881
- "scan for widest needed" seems best [Corbett 198?]

Conflict between compiler's pseudo-machine and actual registers

4. Real Elementary Transcendental Functions:

Argument reduction for log, exponential and trig. functions
- How to simulate $2/\pi$ to infinite precision, or not.

CORDIC algorithms = Argument reduction carried to its limit.
Polynomial, rational and other "best" approximation
- The Remez algorithm, with compensation for roundoff

Interpolation in BIG tables (IBM '370, DEC VAX VMS, ...)
- J. C. P. Miller's trick to suppress roundoff [Gal 198?]

"Kernel function" approach; monotonicity achieved thereby
Special properties of special functions:
- Weak monotonicity: $EXP(X+Y^2) \geq EXP(X)$.
- Cardinal values: $\cos(0) = 1$, $|\cos(x)| \leq 1$.
- Huge arguments: cos(3083975227) = -.00000000007486714572...
- Removable singularities: should $0^0 = 1$?
- Unremovable singularities: $\ln(0) = -\infty$, $\ln(-1)$ is NaN.

Accuracy claims and proofs; the Table-Maker's Dilemma
Testing: W. J. Cody's [1980] ELEFUNT tests
A. Z-S. Liu's [1987] test program

5. Complex Arithmetic

Mixed Real and Complex expressions
Multiply/Divide despite exceptions in concurrent environments
Different approaches to ∞
The sign of Zero: functions like $\sqrt{}$ on slitted domains
Accurate elementary transcendental functions [Kahan 1986]
P-T. Tang's [1987] complex Remez algorithms

6. Exception Handling:

An Exception is not an Error unless it is handled badly.
Why we must neither always ABORT nor always merely CONTINUE.
Retrospective Diagnostics resolve the dilemma.
Let's not get enmeshed in the operating system's kernel.
Concurrent/Parallel/Pipelined machines restrict the options.
  - How can we PAUSE and RESUME during debugging?
∞ and NaN/Indefinite/Reserved operand allow CONTINUE option.
Presubstitution, and some of its applications.
Flags permit recovery from ill-presubstituted exceptions.
Circumventing Over/Underflow in extended products/quotients:
  - Kahan's [1966] implementations on IBM 7094, '360, ...
  - D. Barnett's [1987] on DEC VAX (UNIX), SUN.
The problem of Scope :
  - solved without language support by Apple (SANE) [1986]
  - with minimal language support -
      APL's Localization of System Variables
  - proper language support achieves minimal run-time cost -
      diminish incidence of forgotten SAVE/RESTOREs,
      avoid unnecessary SAVE/RESTORE in leaf-subprograms.
Comparisons with ...
    "ON condition ... " in PL1 and BASIC.
    "drop-through" in ADA.
    "Error-Handlers" in DEC VMS Fortran.
    Signals thrown and caught by UNIX .
    Augmented data-structures for Data-Flow.

7. Numerical Lapses in Standard Programming Languages:

Misguided attempts:
  - to define the semantics of arithmetic,
      - W. S. Brown's model in ADA, Fortran 8x, ANSI C
      - Ambiguous Environmental Inquiries
  - to define things like 1.0/0.0 and $0.0^{0.0}$ as Errors.
  - to ignore integer overflow.
Running roughshod over parentheses (C).
Moving code out of loops despite side-effects.
Other would-be "optimizations" like ...
  - misuse of register variables causing schizophrenia
  - misuse of " 0-x " instead of " -x " , or
    " -(x-y) " instead of " y-x " , and
    " x-y > 0 " instead of " x > y " .
Pascal's fixed array-dimensions, lack of Double-precision,
    inadequate provision for precompiled libraries.
C's awkward multi-subscripted arrays inside procedures.
Fortran's arguments are unprotected - cf. prototypes;
    argument lists have fixed length - cf. optional arguments.
Inability to overload arithmetic operators to introduce ...
  Complex arithmetic, Matrix arithmetic, Interval arithmetic

Awkward treatment of ...
- Library function(function(parameters)); cf. ELSODE
- "Eureka" problem (cf. "Break")
- "Lost search-party" problem (cf. Exception handling)
- Mixed-precision expressions [Corbett 1980]
- Re-running a program with higher precision
- Mixed Real-Complex expressions [Kahan 1987]
- Inexact constants, inaccurate conversions
- Output too wide for its field

What else compiler writers should know about floating-point:
- C. Farnum's [1988] paper