EE 486 : lecture 3, more on IEEE and IEEE vs. Cray arithmetic

M. J. Flynn

Lecture 2, slide 13 update. IEEE format, the basics

- Single: (32b) s+e+m bit layout is 1+8+23 Double: (64b) 1+ 11 +52; also quad
- Extended: (44b) 1+11+32 and (80b) 1+15+64 for register only operations; no hidden one used in an extended format.
- $\beta = 2$ with S + M representation
- Hidden 1 to the left of radix point (1).xxx
- Exponent bias $(2^e/2) -1$ not $2^e/2$

Guard bits and rounding

- The rounding of the result adds A after post normalization

1.	L	G	S

First must combine R and S and then add A to the G bit

A is derived from L, G and S

Action table, A, for RN

L	G	S	Action	A
X	0	0	Exact result, no round	0
X	0	1	Inexact result, but significand rounded	0
0	1	0	Tie case with even L, no rounding	0
1	1	0	Tie case with odd L, round to even	1
X	1	1	Round to nearest	1

Getting the guard bits and rounding

- Before post normalization result should be computed to

X	X.	N’	L’	G’	R’	S’

L’ is the lsb, N’ is the lsb plus 1; G’ is first guard digit, round bit R’ is the second guard bit and S’ in the sticky bit (the OR of all lower order bits.)

Guard bits and rounding

- Before post normalization result can be
  - XX.XX… on +/* 4.0> r >or = 1.0; this requires 0 or 1 right shift.
  - On divide since r_{min} = 1.0/(2-ulp) then 2.0 > r > ½; so one left shift may be required
  - On subtraction, can have massive cancellation so m-1 left shifts can be required.

Finding LGS from L'G'R'S'

- After post normalization:
  - Case 1 no shift G=G' ; S= R'+S'
  - Case 2 one right shift L=N'; G=L'; S=S'+G'+R'
  - Case 3 one left shift L=G'; G=R'; S=S'
  - Case 4 >1 left shift. Can only occur on (a-b) where a,b exponents differ by 0 or 1. So L'G' are left shifted into result and R=0 and S=0 (as R' and S' must also be 0)

A, the rounding action

- A is added to the G bit to produce rounded result.
- Commonly A selects from MUX either result or result+1.
- Must define A for each rounding mode; preferably ***before*** post normalization.

IEEE reserved operands and denormal numbers

S, sign	Biased exp	Significand	Meaning
0	0	0	+ zero
1	0	0	- zero
0/1	0	Not 0	+/- a denormalized number
0	255	0	+ infinity
1	255	0	- infinity
X	255	Not 0	NAN not a number

Denormal numbers

- Min is (1).0 * 2^{-126}
- Denormal has no hidden 1, so (min – ulp) is 0.11111…1 * 2^{-126}; this is the same as 1.111…* 2^{-127} This is the largest denormal.
- Min denormal is 0.0000000…….01*2^{-126}.
- Denormals may have minimum significance if used by subsequent up-scaling operations.

IEEE exceptions

- Five conditions
  - Invalid ops and NANs
  - Overflow
  - Division by 0
  - Underflow
  - Inexact result
- Exceptions are handled by either trap or by disabled trap and continuing operation.

Traps disabled

- Invalid ops: e.g. (sqrt of –5) produces NAN, then the NAN propagates as a valid result, e.g. 5 x NAN =NAN
- Overflows: set to +/- infinity and continue
- Divide by 0: set to +/- infinity
- Underflow: set to denormal and then to 0
- Inexact: continue (used only for integer arithmetic)

Traps enabled

- On overflow/ underflow: adjust char (remove bias on overflow, add k on under) to bring exp into useful range
- On invalid: use the NAN format to provide a useful pointer (user discretion)
- Inexact result: if G and S =0 then exact; otherwise inexact, trap to continue to protect integer computation.

Can FPN satisfy the laws of algebra?

- Idempotency: 1x X=X or X +/- 0 = X for all X..this should be OK; but converse fails so that if (a x X) = a, X need not be 1 and if (a +/- X) = a, then X need not =0.
- Commutation: a + b = b + a and a x b=b x a should be OK, but some designs fail
- Associative: a + (b+c) = (a+b) +c fails
- Distributive: (a+b) x c = a x c + b x c fails

Result sequencing

- In FP, ops should appear to be computed as a serial sequence of sub-ops
  - Operand capture and validation
  - Compare and exponent difference
  - Significand alignment (pre shift)
  - Operation
  - Recomplement
  - LOD and post shift to renormalize
  - Round and final renormalize

Cray (ca.’76): quick and dirty FPN

- Layout is s + e +f = 1 +15 +48 =64b
- β = 2, bias is $2^{15}/2 = 2^{14}$ = 16,384
- Fraction is 0.1xx.., no hidden 1
- Max exp is = 2^{13}-1= 8,191 (not 2^{14}-1)
- Overflow, o, occurs when leading 2 bits of char =11 (exp is 8,192), set flag
- Underflow when leading 2 bits of char = 00 (exp is –8,191); set r to 0, no trap.

More Cray

- But determine o and underflow before (!) post normalization
- No validity test on input operands (except 0 test)
- Only partial build out of multiplier tree
- So (min + s) – min = s where s can be 2^{-48} less that min. Max * 1.0 gives o. Early models didn’t give X * Y = Y * X.

Yet more Cray

- Can produce small=min*2^{-48} as output, but recognized as “0” on input, so small*1.0 = 0
- Can ignore o trap and continue since o char is 11XXXX. But o * 0 = 0.
- Integer ops available; integers have all zero char for both char.

FPNs

- In many applications (eg DSP) you don't need a robust FPN, compromises in FPN can give more effective designs.
- Some FPNs (not so common now) claim IEEE but only give IEEE formatted results
- Finally, there are many versions of IEEE, eg some with extended registers some without.