

Min/Max part 2

IEEE-754r

Roger Golliver

Intel Corp.

May 12, 2005

Agenda

- Answers from last discussion's action item
- Intel® Architecture Min sequences
- Summary
- Backup – errata from March presentation

Min/Max – Special Cases

- We have received statements from three interval experts that min/max special handling of +/-0 is not crucial
 - Prof. Kahan – we need three zeros, i.e. -0, 0, and +0
 - Prof. Kulisch – sign of zero not used in his implementations
 - Prof. Neumaier – as indication of open/closed interval, not useful. It would be useful if all numbers had open above, closed, open below capability, e.g. 1+, 1, 1-
- NaN filtering
 - Prof. Neumaier – uses NaNs to indicate failure, so would prefer min/max to propagate NaNs, not filter them out. But, he can deal with an definition, as long as it is documented and widely adopted.

c99 Min – Filtering NaNs only

Itanium®-2 version – three instructions:

```
static inline fmin ( x, y ) { double z;
    if( isnan( y ) )      // fclass.s1 (NaN, y)
        z = x;          // fmov    z = x
    else
        z = x < y ? x : y; // fmin.s1 z = x, y
    return z;
}
```

IA-32/x87 version – four instructions:

```
static inline fmin ( x, y ) { double z;
    eflags = fucomi( y, y); // doesn't set invalid
    y = fcmovu ( eflags, x, y); // when y is a NaN replace with x
    eflags = fucomi( x, y ); // doesn't set invalid
    z = fcmovbe( eflags, x, y); // only returns NaN if both x and y are NaNs
    return z;
}
```

IA-32/SSE/SSE2 version – eight instructions per pair (or quad):

```
static inline fmin ( x, y ) { xmm128 mx, my, mxy, x1, x2, x3, y1, y2, y3, z1, z2, z3;
    mx = cmpd.ne( x, x ); // ones where x is a NaN, won't set invalid
    my = cmpd.ne( y, y ); // ones where y is a NaN, won't set invalid
    mxy = andpd ( mx, my ); // ones where x and y are NaNs
    x1 = andnpd( mx, x ); // zero NaNs in x
    y1 = andnpd( my, y ); // zero NaNs in y
    x2 = andpd ( my, x1 ); // x2 is x1 where y is a NaN (either x or zeros)
    y2 = andpd ( mx, y1 ); // y2 is y1 where x is a NaN (either y or zeros)
    x3 = orpd ( x1, y2 ); // x3 is x1 with y2 (either y or zeros) where x is a NaN
    y3 = orpd ( y1, x2 ); // y3 is y1 with x2 (either x or zeros) where y is a NaN
    z1 = andpd ( mxy, x ); // extract NaN from x where both are NaNs
    z2 = minpd ( x1, y1 ); // min of x and y, won't set invalid, zero's where both are NaNs
    z3 = orpd ( z1, z2 ); // merge NaN, if any, into final answer
    return z3;
}
```

Itanium®-2 min case analysis

left arg	right arg	fclass (NaN,r)	T:: fmov: z = l	F:: fmin: z = l < r ? l : r
+non_0_L	NaN	T	F: +non_0_L	
+0	NaN	T	F: +0	
-0	NaN	T	F: -0	
-non_0_L	NaN	T	F: -non_0_L	
NaN_L	NaN_R	T	F: NaN_L Non-symmetric	
NaN	+non_0_R	F		F: +non_0_R
NaN	+0	F		F: +0
NaN	-0	F		F: -0
NaN	-non_0_R	F		F: -non_0_R
non_NaN	non_NaN	F		Min(l,r) Non-sym for 0

x87 case analysis

left arg	right arg	fucomi r,r	fcmovu l,r	fucomi l,r	fcmovbe r,l
other_l	other_r	ordered	other_r	ordered	Min(other_l, other_r)
other_l	nan_r	unordered	other_l	ordered	Min(other_l, other_l)
nan_l	other_r	ordered	other_r	ordered	Min(other_l, other_r)
nan_l	nan_r	unordored	other_l	unordered	Min(nan_l, nan_r)

SSE/SSE2 case analysis

left arg	right arg	mx	my	mxy	x1	y1	x2	y2	x3	y3	z1	z2	z3
oL	oR	0	0	0	oL	oR	0	0	oL	oR	0	min	min
oL	nR	0	1	0	oL	0	oL	0	oL	oL	0	min	min
nL	oR	1	0	0	0	oR	0	oR	oR	oR	0	min	min
nL	nR	1	1	1	0	0	0	0	0	0	nL	0	nL

nL, nR – nan for left/right argument

oL, oR – other than NaN for left/right argument

Min – min between two ordered arguments(x3,y3)

Summary

- Cost of doing special NaN handling and special 0 handling is high and justification seems to be symmetry not utility
- Cost of doing c99 special NaN handling is less, but still more than I think users will want for default min usage
 - 3x for Itanium®-2
 - 2x for x87
 - 12x for SSE/SSE2
- I feel a better definition of min for IEEE-754r would be:
 - Default language binding to a “fused compare and conditional move” definition of min (something all ISAs can do today at no additional cost)
 - Additionally a c99 fmin version; accessible via a non-language default name (so only those that need the special semantics pay the cost)

Backup – errata

(I made a mistake in March)

- This Itanium®-2 sequence for screening NaNs and handling -0 was in error:

```
static inline min(x,y) {
    if ( iszero(y)&& signbit(y) ) // fclass neg&&zero
        z = x < y ? x : y; // fmin
    else
        z = y < x ? y : x; // fmin
    return z;
}
```

case analysis which caught error (using previous page's definition)

left arg	right arg	fclass (-0,r)	fmin: l<r ? l : r	fmin: r<l ? r : l
+non_0_L	-0	T	F: -0	
+0	-0	T	F: -0	
-0	-0	T	F: -0	
-non_0_L	-0	T	T: -non_0_L	
NaN	-0	T	F: -0	
+non_0_L	+0	F		F: +non_0_R
+0	+0	F		F: +0
-0	+0	F		F: -0
-non_0_L	+0	F		T: -non_0_R
NaN	+0	F		F: NaN should be +0