

Floating-Point Reference Sheet for Intel® Architecture

<https://software.intel.com/en-us/articles/floating-point-reference-sheet-for-intel-architecture> (v2.12) | Michael Ferry

Binary Format Floating-Point Number

Sign	Biased Exponent	Significand					
s	E	x ₁	x ₂	x ₃	... x _{p-1}	x _p	
MSB		J-bit	Fraction				LSB

$$= \begin{cases} (-1)^s \times x_1.x_2x_3 \dots x_{p-1}x_p \times 2^{E-B}, & \text{if normal} \\ (-1)^s \times x_1.x_2x_3 \dots x_{p-1}x_p \times 2^{e_{min}}, & \text{if denormal} \end{cases}$$

- Sign bit is s = 0 for '+', and s = 1 for '-' (also refer to 's' as 'sign')
- Unbiased exponent is e = E - B - x₁ + 1 for nonzero finite numbers
- For standard formats, x₁ equals (E ≠ 0) and is implicit
- For NaNs, the payload is the bit string from x₃ to x_p

Floating-Point Classes, Encodings, and Parameters

	Standard Formats*				Extended Format*				Non-Std*			
	E	J	Fraction	Values	Half (16b)	Single (32b)	Double (64b)	Quad (128b)	x87 (80b) t***	Bfloat (16b)		
Zero	00...00	0	00...00	+Zero	0000	0000 0000	0000 0000 0000 0000	0000 0000 ... 0000	0000 0000 ... 0000	0000		
Denormal	00...01	0	00...01	+D _{min}	0001	0000 0001	0000 0000 0000 0001	0000 0000 ... 0001	0000 0000 ... 0001	0001		
	11...11		11...11	+D _{max}	03fff	007f ffff	000f ffff ffff ffff	0000 ffff ... ffff	0000 7fff ... ffff	007f		
Normal	00...01	0	00...00	+N _{min}	0400	0080 0000	0010 0000 0000 0000	0001 0000 ... 0000	0001 8000 ... 0000	0080		
	11...10		11...11	+N _{max}	3c00	3f80 0000	3ff0 0000 0000 0000	3fff 0000 ... 0000	3fff 8000 ... 0000	3f80		
Infinity	00...00	1	00...00	+Infinity	7c00	7f80 0000	7ff0 0000 0000 0000	7fff 0000 ... 0000	7fff 8000 ... 0000	7f80		
sNaN	00...01		00...01	"+"sNaN	7c01	7f80 0001	7ff0 0000 0000 0001	7fff 0000 ... 0001	7fff 8000 ... 0001	7f81		
	01...11		01...11	"-"sNaN	7dff	7fbf ffff	7ff7 ffff ffff ffff	7fff 7fff ... ffff	7fff bfff ... ffff	7fbf		
qNaN	10...00	10...00	R Ind**	fe00	ffc0 0000	fff8 0000 0000 0000	ffff 8000 ... 0000	ffff c000 ... 0000	ffc0			
qNaN	11...11	11...11	"+"qNaN	7e00	7fc0 0000	7ff8 0000 0000 0000	7fff 8000 ... 0000	7fff c000 ... 0000	7fc0			
			"-"qNaN	7fff	7fff ffff	7fff ffff ffff ffff	7fff ffff ... ffff	7fff ffff ... ffff	7fff			
Field												
# of Bits												
Exp. bias (B)												
E _{min} : E _{max}												

* All examples are in little endian byte order ** R Ind (Real Indefinite), a qNaN, must have sign bit s = 1 and payload = 00...00
 *** Two additional classes exist for x87 80-bit format: pseudo-denormal (E = 0, J = 1) and unsupported (E ≠ 0, J = 0)

Operation-Specific Results and Faults for Typical Intel® SSE or Intel® AVX Scalar Instructions

- If DAZ = 1, denormal inputs are replaced with appropriately signed zeros
- Q(X) (Quiet(X)) sets the most significant fraction bit of X (x₂) to 1
- For more details on exception priorities and unmasked behavior, see flowchart on next page
- NaN payload's least significant bits are zero-extended or truncated to fit the destination

NaN Behavior: Add/Sub/Mul/Div		Src2		
		sNaN	qNaN	Other
Src1	sNaN	Q(Src1)	Q(Src1)	Q(Src1)
	qNaN	Src1	Src1	Src1
	Other	Q(Src2)	Src2	op-specific

Non-NaN X * Y sign = X.s ^ Y.s		Y			
		Infinity	Normal	Denormal	Zero
X	Infinity	Infinity	Infinity	Infinity	R Ind
	Normal	Infinity	X * Y	X * Y	0.0
	Denormal	Infinity	X * Y	X * Y	0.0
	Zero	R Ind	0.0	0.0	0.0

Non-NaN X / Y sign = X.s ^ Y.s		Y			
		Infinity	Normal	Denormal	Zero
X	Infinity	R Ind	Infinity	Infinity	Infinity
	Normal	0.0	X / Y	X / Y	Infinity
	Denormal	0.0	X / Y	X / Y	Infinity
	Zero	0.0	0.0	0.0	R Ind

NaN Behavior: FMA (X*Y + Z)		Z		
		sNaN	qNaN	Other
X,Y	sNaN, sNaN	Q(X)	Q(X)	Q(X)
	sNaN, qNaN	Q(X)	Q(X)	Q(X)
	sNaN, Other	Q(X)	Q(X)	Q(X)
	qNaN, sNaN	X	X	X
	qNaN, qNaN	X	X	X
	qNaN, Other	X	X	X
	Other, sNaN	Q(Y)	Q(Y)	Q(Y)
	Other, qNaN	Y	Y	Y
	Other, Other	Q(Z)	Z	X*Y+Z

Non-NaN X + Y [X - Y = X + (-Y)]		Y					
		+Infinity	-Infinity	Normal	Denormal	+Zero	-Zero
X	+Infinity	X	R Ind	X	X	X	X
	-Infinity	R Ind	X	X	X	X	X
	Normal	Y	Y	X+Y*	X+Y*	X	X
	Denormal	Y	Y	X+Y*	X+Y*	X	X
	+Zero	Y	Y	Y	Y	+0.0	0.0*
	-Zero	Y	Y	Y	Y	0.0*	-0.0

* If X + Y is exactly 0, sign bit s equals (RC == -INF)

Sqrt(X)	
sNaN	Q(X)
qNaN	X
-Infinity	X
Infinity	R Ind
+Normal	Sqrt(X)
-Normal	R Ind
+Denormal	Sqrt(X)
-Denormal	R Ind
Zero	X

Convert(X)		Fp2Int(X)	Fp2Fp(X)	Int2Fp(X)
X	sNaN	Int Ind	Q(X)	N/A
	qNaN	Int Ind	X	N/A
	Infinity	Int Ind	X	N/A
	Normal	Fp2Int(X)	Fp2Fp(X)	Int2Fp(X)
	Denormal	Fp2Int(X)	Fp2Fp(X)	N/A
Zero	0	X	+0	

Int Ind (Integer Indefinite) is defined to be the bit string 10...00

* If Fp2Int(X) is not representable in dest format, raise I

Non-NaN X*Y+Z [XY + Z]		Z					
		+Infinity	-Infinity	Normal	Denormal	+Zero	-Zero
XY	R Ind	R Ind	R Ind	R Ind	R Ind	R Ind	R Ind
	+Infinity	XY	R Ind	XY	XY	XY	XY
	-Infinity	R Ind	XY	XY	XY	XY	XY
	Normal	Z	Z	XY+Z**	XY+Z**	XY	XY
	Denormal	Z	Z	XY+Z**	XY+Z**	XY	XY
	+Zero	Z	Z	Z	Z	+0.0	0.0**
	-Zero	Z	Z	Z	Z	0.0**	-0.0

* If X or Y is Denormal and X*Y+Z does not raise I, raise D

** If XY + Z is exactly 0, sign bit s equals (RC == -INF)

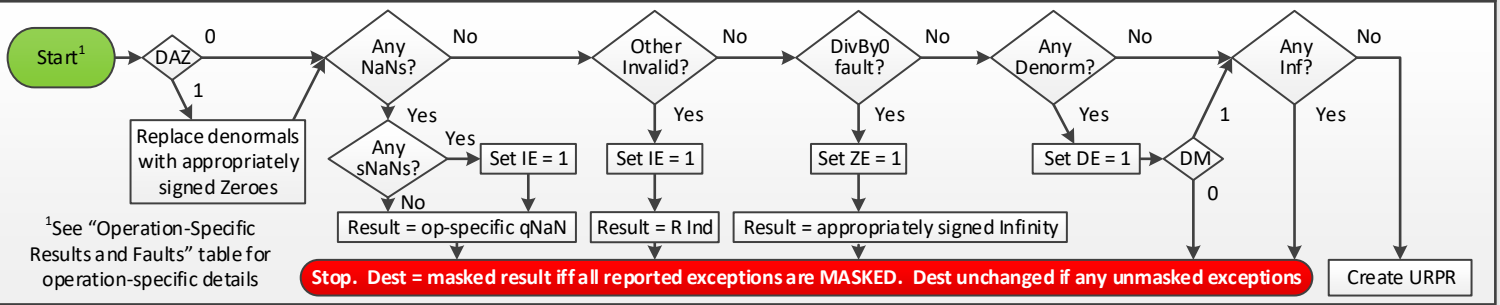
Control and Status Words

x87	FPCW	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	FPSW	B	C3		X	RC		PC		ES	SF						
	MXCSR	FTZ	RC														
					P	U	O	Z	D	I							

- *E, *M: Exceptions and Masks Precision (P), Underflow (U), Overflow (O), Divide-by-Zero (Z), Denormal Inputs (D), Invalid Inputs (I)
- RC: Round Control RoundTiesToEven / RoundToNearestEven (RNE), RoundTowardsNegative (-INF), RoundTowardsPositive (+INF), RoundTowardZero (RTZ)
- PC: Precision Control Single Precision (SP), Double Precision (DP), Double Extended Precision (DEP)
- Underflow / Denormals Flush to Zero (FTZ), Denormals Are Zero (DAZ)

Flowchart for a Typical Intel® SSE or Intel® AVX Floating-Point Scalar Instruction

Precomputation & Fault Handling



Unnormalized Reduced Precision Result (URPR) $URPR = (-1)^s \times x_0 x_1 \dots x_{p-1} L G R S \times 2^{exp}$ significand $\in [0,4)$ exp unbounded

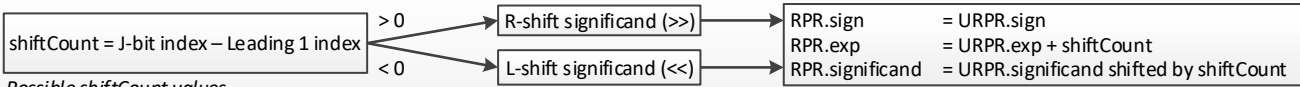
Theoretical: Compute the Infinitely Precise Result (IPR); if not representable, choose one of the two nearest representable FP numbers using IEEE 754 rounding process.
 Practical effect: we must usually compute URPR instead. The URPR is formed from the terminating representation of the IPR if one exists (ex.: 10.0₂ vs. 01.111...₂).

Operation	Input Manipulation	Leading 1	URPR.exp	Guard	Round	Sticky
X + Y	True Add	Denormalize smaller number (R-shift) to make exponents equal, if required	x_0 or x_1	$\max(X.exp, Y.exp)$	N/A	IPR. x_{p+1} OR (IPR. $x_{p+2}, IPR.x_{p+3}, \dots$)
X - Y	True Sub ²		$x_1 - x_p$, or URPR = 0.0			
XY + Z	FMTTrueAdd	Denormalize smaller of XY or Z (R-shift) to make exponents equal, if required	x_0 or x_1			
XY - Z	FMTTrueSub ²		$x_1 - x_{2p-1}$, or URPR = 0.0			
X × Y	Multiply	None	x_0 or x_1	X.exp + Y.exp		
\sqrt{x}	Sqrt	L-shift significand to make exponent even, if required	x_1	(X.exp + 1) >> 1		
X / Y	Divide	None	x_1 or x_2	X.exp - Y.exp	IPR. x_{p+1}	IPR. x_{p+2} OR (IPR. $x_{p+3}, IPR.x_{p+4}, \dots$)

²A heterogeneous sub (Ex: homogeneous FMA true subtraction) requires a set of guard bits

Reduced Precision Result (RPR) $RPR = \pm 0$ or $(-1)^s \times 1.x_2 \dots x_{p-1} L R S \times 2^{exp}$ significand $\in \{0\} \cup [1,2)$ exp unbounded

Normalize the URPR: shift the significand until leading 1 is in the J-bit position.



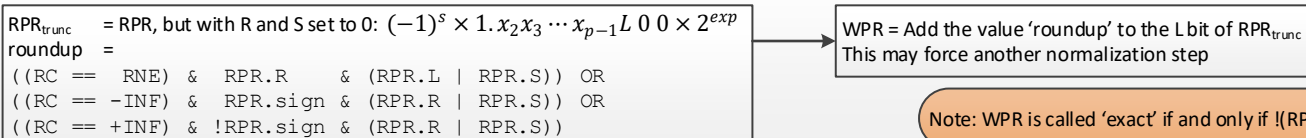
Possible shiftCount values

True Add	FMTTrueAdd	Multiply	True Sub	FMTTrueSub	Sqrt	Divide
	[0,1]		[-p,0]	[-(2p-1),0]	0	[-1,0]

Note: When shifting right, don't discard bits!
 $RPR.S = \text{OR}(URPR.x_{\text{stickyIndex}} \dots URPR.x_{\text{stickyIndex} - \text{shiftCount}})$

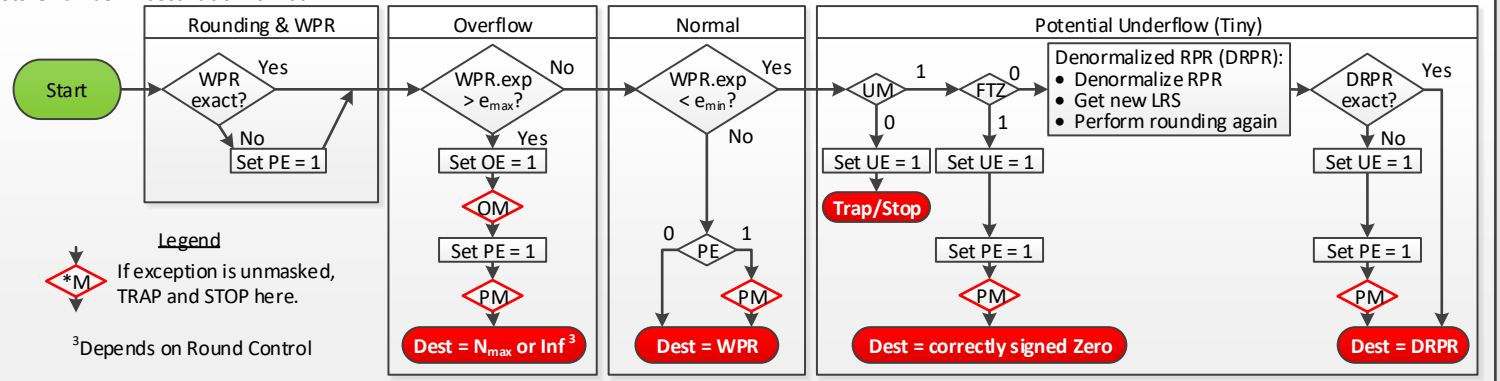
Rounding & Working Precision Result (WPR) $WPR = \pm 0$ or $(-1)^s \times 1.x_2 x_3 \dots x_p \times 2^{exp}$ significand $\in \{0\} \cup [1,2)$ exp unbounded

Fit the significand into p bits, performing rounding and exponent adjustment if necessary; allow unbounded exponent.



Stored Result & Trap Handling Stored Result = $\pm Inf$ or $(-1)^s \times x_1.x_2 x_3 \dots x_p \times 2^{exp}$ significand $\in [0,2)$ finite, nonzero exp $\in [e_{min}, e_{max}]$

Store number in destination format.



Intel disclaims all express and implied warranties, including without limitation, the implied warranties of merchantability, fitness for a particular purpose, and non-infringement, as well as any warranty arising from course of performance, course of dealing, or usage in trade.

Intel is a trademark of Intel Corporation in the U.S. and/or other countries.

Copyright © 2018, Intel Corporation.