Floating-Point Reference Sheet for Intel® Architecture

https://software.intel.com/en-us/articles/floating-point-reference-sheet-for-intel-architecture (v2.13)

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*

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

- Sign bit is s = 0 for '+', and s = 1 for '−' (also refer to 's' as 'sign')
- Unbiased exponent is e = E – B – x_1 + 1 for nonzero finite numbers
- For standard formats, x_1 equals (E ≠ 0) and is implicit
- For NaNs, the payload is the bit string from x_3 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 **0**000 … 0000	0000
Denormal			00…01 ↔ 11…11	+D_{min}	0001	0000 0001	0000 0000 0000 0001	0000 0000 … 0001	0000 **0**000 … 0001	0001
				+D_{max}	03ff	007f ffff	000f ffff ffff ffff	0000 ffff … ffff	000**0** **7**fff … ffff	007f
Normal	00…01 ↔ 11…10	1	00…00 ↔ 11…11	+N_{min}	0400	0080 0000	0010 0000 0000 0000	0001 0000 … 0000	000**1** **8**000 … 0000	0080
				+One	3c00	3f80 0000	3ff0 0000 0000 0000	3fff 0000 … 0000	3fff **8**000 … 0000	3f80
				+N_{max}	7bff	7f7f ffff	7fef ffff ffff ffff	7ffe ffff … ffff	7ff**e** **f**fff … ffff	7f7f
Infinity	11…11		00…00	+Infinity	7c00	7f80 0000	7ff0 0000 0000 0000	7fff 0000 … 0000	7ff**f** **8**000 … 0000	7f80
sNaN			00…01 ↔ 01…11	"+"sNaN	7c01	7f80 0001	7ff0 0000 0000 0001	7fff 0000 … 0001	7fff **8**000 … 0001	7f81
					7dff	7fbf ffff	7ff7 ffff ffff ffff	7fff 7fff … ffff	7fff **b**fff … ffff	7fbf
qNaN			10…00 ↔ 11…11	R Ind**	fe00	ffc0 0000	fff8 0000 0000 0000	ffff 8000 … 0000	ffff **c**000 … 0000	ffc0
				"+"qNaN	7e00	7fc0 0000	7ff8 0000 0000 0000	7fff 8000 … 0000	7fff **c**000 … 0000	7fc0
					7fff	7fff ffff	7fff ffff ffff ffff	7fff ffff … ffff	7fff **f**fff … ffff	7fff

	Half (16b)				Single (32b)				Double (64b)				Quad (128b)				x87 (80b)				Bfloat (16b)			
Field	s	E	J	F	s	E	J	F	s	E	J	F	s	E	J	F	s	E	**J**	F	s	E	J	F
# of Bits	1	5	0	10	1	8	0	23	1	11	0	52	1	15	0	112	1	15	**1**	63	1	8	0	7
Exp. bias (B)	0x0f (15)				0x7f (127)				0x3ff (1023)				0x3fff (16383)				0x3fff (16383)				0x7f (127)			
e_{min} : e_{max}	−14		15		-126		127		-1022		1023		−16382		16383		−16382		16383		-126		127	

* 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 (x_2) 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)	I	Q(Src1)	I	Q(Src1)	I
	qNaN	Src1	I	Src1		Src1	
	Other	Q(Src2)	I	Src2		*op-specific*	

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

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

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

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

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

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

Convert(X)		Fp2Int(X)		Fp2Fp(X)		Int2Fp(X)	
X	sNaN	Int Ind	I	Q(X)	I	N/A	
	qNaN	Int Ind	I	X		N/A	
	Infinity	Int Ind	I	X		N/A	
	Normal	Fp2Int(X)	*	Fp2Fp(X)		Int2Fp(X)	
	Denormal	Fp2Int(X)		Fp2Fp(X)	D	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	I	R Ind	I	R Ind	I	R Ind	I	R Ind	I	R Ind	I
	+Infinity	XY	*	R Ind	I	XY	*	XY	D	XY	*	XY	*
	-Infinity	R Ind	I	XY	*	XY	*	XY	D	XY	*	XY	*
	Normal	Z	*	Z	*	XY+Z**	*	XY+Z**	D	XY	*	XY	*
	Denormal	Z	D	Z	D	XY+Z**	D	XY+Z**	D	XY	D	XY	D
	+Zero	Z	*	Z	*	Z	*	Z	D	+0.0	*	0.0**	*
	-Zero	Z	*	Z	*	Z	*	Z	D	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

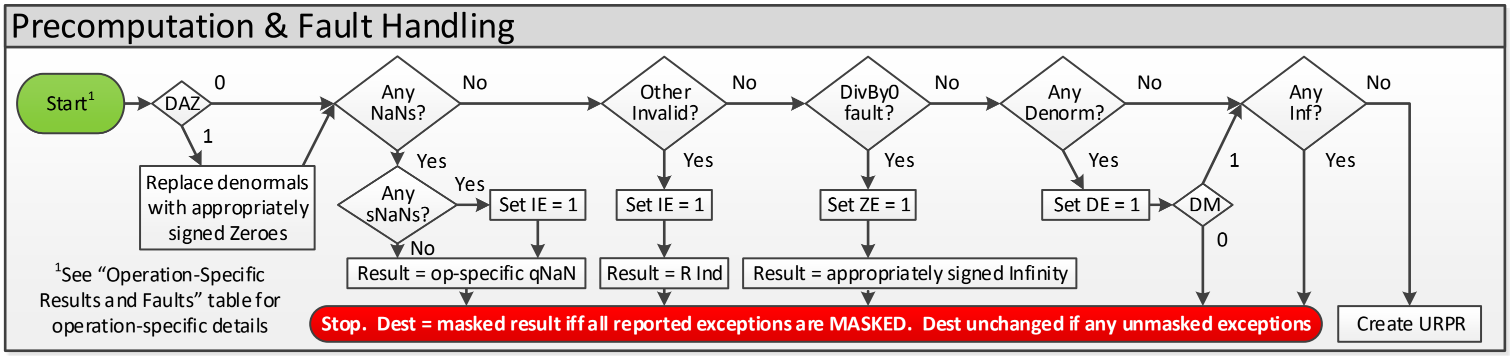
		15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
x87	FPCW				X	RC		PC						Masks			
	FPSW	B	C3	Top			C2	C1	C0	ES	SF			Exceptions			
SSE, AVX	MXCSR	FTZ	RC			Masks					DAZ			Exceptions			
					P	U	O	Z	D	I		P	U	O	Z	D	I

	00	01	10	11
RC	RNE	-INF	+INF	RTZ
PC	SP		DP	DEP

*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



Start[1]

DAZ

0

1

Replace denormals with appropriately signed Zeroes

Any NaNs?

No

Yes

Any sNaNs?

Yes

No

Set IE = 1

Result = op-specific qNaN

Other Invalid?

No

Yes

Set IE = 1

Result = R Ind

DivBy0 fault?

No

Yes

Set ZE = 1

Result = appropriately signed Infinity

Any Denorm?

No

Yes

Set DE = 1

DM

1

0

Any Inf?

No

Yes

Create URPR

Stop. Dest = masked result iff all reported exceptions are MASKED. Dest unchanged if any unmasked exceptions

[1]See "Operation-Specific Results and Faults" table for operation-specific details

Unnormalized Reduced Precision Result (URPR)

URPR =$(-1)^s \times x_0x_1.x_2 \cdots x_{p-1} L\ G\ R\ S \times 2^{exp}$ | significand ∈ [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.: 10.0_2 vs. $01.111..._2$).

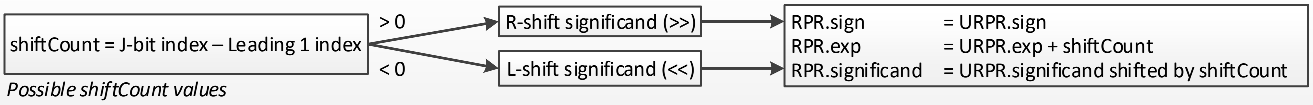
Operation		Input Manipulation	Leading 1	URPR.exp	Guard	Round	Sticky
X + Y	True Add	Denormalize smaller number (R-shift) to make exponents equal, if required	x_0 or x_1	max(X.exp,Y.exp)	N/A	IPR.x_{p+1}	OR(IPR.x_{p+2},IPR.x_{p+3},...)
X − Y	True Sub[2]		$x_1 - x_p$, or URPR = 0.0				
XY + Z	FMTrueAdd	Denormalize smaller of XY or Z (R-shift) to make exponents equal, if required	x_0 or x_1	max(XY.exp,Z.exp)			
XY − Z	FMTrueSub[2]		$x_1 - x_{2p-1}$, or URPR = 0.0				
X × Y	Multiply	None	x_0 or x_1	X.exp + Y.exp			
√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},...)

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

Reduced Precision Result (RPR)

RPR = ±0 or $(-1)^s \times 1.x_2 \cdots x_{p-1} L\ R\ S \times 2^{exp}$ | significand ∈ {0} ∪ [1,2) | exp unbounded

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



shiftCount = J-bit index – Leading 1 index

> 0

< 0

R-shift significand (>>)

L-shift significand (<<)

RPR.sign = URPR.sign
RPR.exp = URPR.exp + shiftCount
RPR.significand = URPR.significand shifted by shiftCount

Possible shiftCount values

True Add	FMTrueAdd	Multiply	True Sub	FMTrueSub	Sqrt	Divide
[0,1]			[−p, 0]	[−(2p−1), 0]	0	[−1, 0]

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

Rounding & Working Precision Result (WPR)

WPR = ± 0 or $(-1)^s \times 1.x_2x_3 \cdots x_p \times 2^{exp}$ | significand ∈ {0} ∪ [1,2) | exp unbounded

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

RPR_{trunc} = RPR, but with R and S set to 0: $(-1)^s \times 1.x_2x_3 \cdots x_{p-1} L\ 0\ 0 \times 2^{exp}$
roundup =

```
((RC  ==  RNE)  &   RPR.R     &  (RPR.L  |  RPR.S))  OR
((RC  == -INF)  &   RPR.sign  &  (RPR.R  |  RPR.S))  OR
((RC  == +INF)  &  !RPR.sign  &  (RPR.R  |  RPR.S))
```

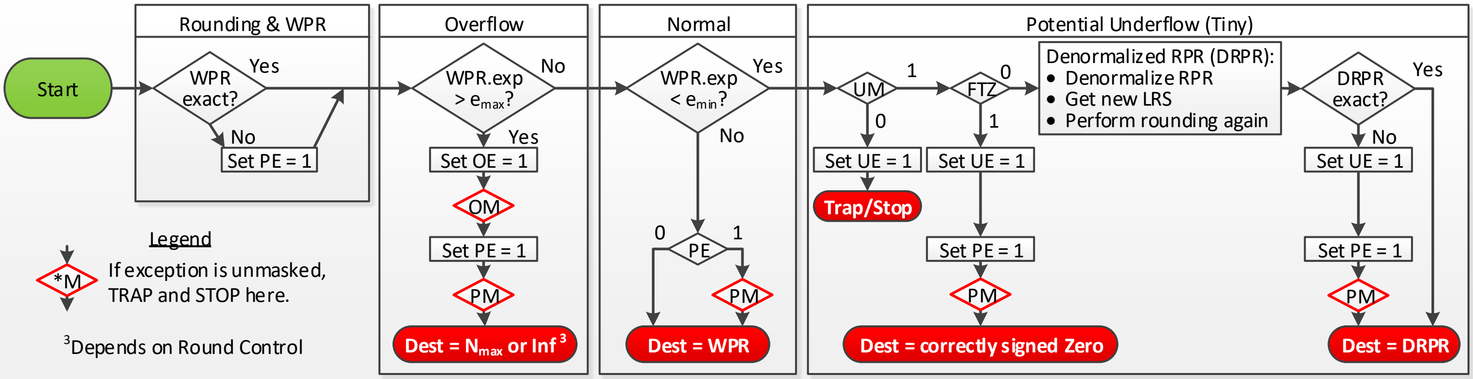
WPR = Add the value 'roundup' to the L bit of RPR_{trunc}
This may force another normalization step

Note: WPR is called 'exact' if and only if !(RPR.R | RPR.S)

Stored Result & Trap Handling

Stored Result = ±Inf or $(-1)^s \times x_1.x_2x_3 \cdots x_p \times 2^{exp}$ | significand ∈ [0,2) | finite, nonzero exp ∈ [e_{min} ,e_{max}]

Store number in destination format.



Start

Rounding & WPR

WPR exact?

Yes

No

Set PE = 1

Overflow

WPR.exp > e_{max}?

No

Yes

Set OE = 1

OM

Set PE = 1

PM

Dest = N_{max} or Inf[3]

Normal

WPR.exp < e_{min}?

Yes

No

PE

0

1

PM

Dest = WPR

Potential Underflow (Tiny)

UM

1

0

Set UE = 1

Trap/Stop

FTZ

0

1

Set UE = 1

Set PE = 1

PM

Dest = correctly signed Zero

Denormalized RPR (DRPR):
- Denormalize RPR
- Get new LRS
- Perform rounding again

DRPR exact?

Yes

No

Set UE = 1

Set PE = 1

PM

Dest = DRPR

Legend

*M If exception is unmasked, TRAP and STOP here.

[3]Depends on Round Control

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.