PowerPC Assembly-Language Numerics Reference

This appendix provides a reference for the numeric implementation in PowerPC assembly language. It summarizes the data formats available, how to determine the floating-point class for a value, the FPSCR, instructions that access the FPSCR, and instructions that perform floating-point operations and the exceptions they might raise.

Floating-Point Data Formats

Figure F-1 Floating-point data formats

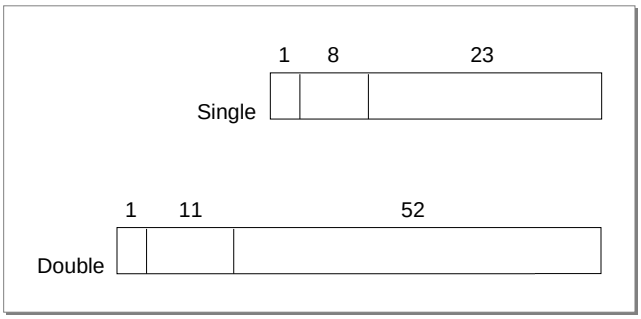

Table F-1 Interpreting floating-point values

If biased† exponent *e* is:	And fraction *f* is:	Then value *v* is:	And class of *v* is:
$0 < e < max$ ‡	(any)	$v = (-1)^s \times 2^{(e-bias)} \times (1.f)$	Normalized number
$e = 0$	$f \neq 0$	$v = (-1)^s \times 2^{minexp} \times (0.f)$ §	Denormalized number
$e = 0$	$f = 0$	$v = (-1)^s \times 0$	Zero
$e = max$	$f = 0$	$v = (-1)^s \times \infty$	Infinity
$e = max$	$f \neq 0$	v = NaN	NaN

† *bias* = 127 for single format, 1023 for double format.
‡ *max* = 255 for single format, 2047 for double format.
§ *minexp* = –126 for single format, –1022 for double format.

Floating-Point Status and Control Register

Table F-2 Bit assignments for FPSCR fields

FPSCR field	Bit	Meaning if set
0	0	Exception summary
	1	Exception enable summary
	2	Invalid-operation exception summary
	3	Overflow exception
1	4	Underflow exception
	5	Divide-by-zero exception
	6	Inexact exception
	7	Invalid-operation exception; signaling NaN as input
2	8	Invalid-operation exception; $\infty - \infty$
	9	Invalid-operation exception; ∞ / ∞
	10	Invalid-operation exception; $0 / 0$
	11	Invalid-operation exception; $0 \times \infty$
3	12	Invalid-operation exception; comparison operation
	13	Fraction field rounded
	14	Fraction field inexact
	15	Class descriptor
4	16	< or < 0
	17	> or > 0
	18	= or = 0
	19	Unordered or NaN
5	20	Reserved
	21	Invalid-operation exception; software request (not implemented in MPC601)
	22	Invalid-operation exception; square root (not implemented in MPC601)
	23	Invalid-operation exception; convert-to-integer operation

continued

Table F-2 Bit assignments for FPSCR fields (continued)

FPSCR field	Bit	Meaning if set
6	24	Invalid-operation exception enable/disable
	25	Overflow exception enable/disable
	26	Underflow exception enable/disable
	27	Divide-by-zero exception enable/disable
7	28	Inexact exception enable/disable
	29	Reserved
	30	Rounding direction
	31	Rounding direction

Table F-3 Rounding direction bits in the FPSCR

Modes	Bits	
	30	31
To-nearest	0	0
Upward	1	0
Downward	1	1
Toward-zero	0	1

Table F-4 Class and sign inquiry bits in the FPSCR

Class/sign	Bits				
	15	16	17	18	19
+0	0	0	0	1	0
−0	1	0	0	1	0
Positive normalized number	0	0	1	0	0
Negative normalized number	0	1	0	0	0
Positive denormalized number	1	0	1	0	0
Negative denormalized number	1	1	0	0	0
+∞	0	0	1	0	1
−∞	0	1	0	0	1
Quiet NaN	1	0	0	0	1

Instructions

Note

Throughout the tables that follow, in the Exceptions column, I = invalid; X = inexact; O = overflow; U = underflow; D = divide by zero. In the Instructions column, * = append dot (.) to instruction name to update CR1. ◆

Table F-5 FPSCR instructions

Instruction	Description	SRC	DST	Exceptions
`mcrfs` *DST*, *SRC*	*DST* ← (*SRC*)	FPSCR field	CR field	- - - - -
`mffs`* *DST*	*DST* ← (FPSCR)	FPSCR	FPR	- - - - -
`mtfsf`* *DST*, *SRC*	*DST* ← *SRC*	FPR	FPSCR field	- - - - -
`mtfsfi`* *DST*, *n*	*DST* ← *n*	16-bit signed int	FPSCR field	- - - - -
`mtfsb1`* *DST*	*DST* ← 1	—	FPSCR bit	- - - - -
`mtfsb0`* *DST*	*DST* ← 0	—	FPSCR bit	- - - - -

Table F-6 Load instructions

Instruction	Description†	SRC	DST	Exceptions
`lfd` *DST*, *n*(*GPR*)	*DST* ← (*n* + (*GPR*))	Memory	FPR	- - - - -
`lfdu` *DST*, *n*(*GPR*)	*DST* ← (*n* + *(GPR)*) *GPR* ← *n* + (*GPR*)	Memory	FPR	- - - - -
`lfdx` *DST*, *GPR1*, *GPR2*	*DST* ← ((*GPR1*) + (*GPR2*))	Memory	FPR	- - - - -
`lfdux` *DST*, *GPR1*, *GPR2*	*DST* ← ((*GPR1*) + (*GPR2*)) *GPR1* ← (*GPR1*) + (*GPR2*)	Memory	FPR	- - - - -
`lfs` *DST*, *n*(*GPR*)	*DST* ← (*n* + (*GPR*)) ‡	Memory	FPR	- - - - -
`lfsu` *DST*, *n*(*GPR*)	*DST* ← (*n* + *(GPR)*) *GPR* ← *n* + (*GPR*)‡	Memory	FPR	- - - - -
`lfsx` *DST*, *GPR1*, *GPR2*	*DST* ← ((*GPR1*) + (*GPR2*))‡	Memory	FPR	- - - - -
`lfsux` *DST*, *GPR1*, *GPR2*	*DST* ← ((*GPR1*) + (*GPR2*)) *GPR1* ← (*GPR1*) + (*GPR2*) ‡	Memory	FPR	- - - - -

† If *GPR* or *GPR1* is 0, the value 0 is used instead of the contents of the register.
‡ Converts single to double format automatically.

Table F-7 Store instructions

Instruction	Description†	SRC	DST	Exceptions
`stfd` *SRC*, *n*(*GPR*)	*n* + (*GPR*)← (*SRC*)	FPR	Memory	- - - - -
`stfdu` *SRC*, *n*(*GPR*)	*n* + (*GPR*)← (*SRC*) *GPR* ← *n* + (*GPR*)	FPR	Memory	- - - - -
`stfdx` *SRC*, *GPR1*, *GPR2*	(*GPR1*) + (*GPR2*)← (*SRC*)	FPR	Memory	- - - - -
`stfdux` *SRC*, *GPR1*, *GPR2*	(*GPR1*) + (*GPR2*)← (*SRC*) *GPR1* ← (*GPR1*) + (*GPR2*)	FPR	Memory	- - - - -
`stfs` *SRC*, *n*(*GPR*)	*n* + (*GPR*)← (*SRC*) ‡	FPR	Memory	- - - - -
`stfsu` *SRC*, *n*(*GPR*)	*n* + (*GPR*)← (*SRC*) *GPR* ← *n* + (*GPR*)‡	FPR	Memory	- - - - -
`stfsx` *SRC*, *GPR1*, *GPR2*	(*GPR1*) + (*GPR2*)← (*SRC*) ‡	FPR	Memory	- - - - -
`stfsux` *SRC*, *GPR1*, *GPR2*	(*GPR1*) + (*GPR2*)← (*SRC*) *GPR1* ← (*GPR1*) + (*GPR2*) ‡	FPR	Memory	- - - - -

† If *GPR* or *GPR1* is 0, the value 0 is used instead of the contents of the register.
‡ Converts double to single automatically.

Table F-8 Conversions to integer format

Instruction	Description	SRC	DST	Exceptions
`fctiw*` *DST*, SRC	*DST* ← (*SRC*) rounded to 32-bit int	FPR	GPR	`I X - - -`
`fctiwz*` *DST*, SRC	*DST* ← (*SRC*) truncated to 32-bit int	FPR	GPR	`I X - - -`

Table F-9 Conversions from double to single format

Instruction	Description	SRC	DST	Exceptions
`frsp*` *DST*, SRC	*DST* ← (*SRC*) rounded to single format	FPR	FPR	`I X O U -`

Table F-10 Comparison instructions

Instruction	Description	SRC	DST	Exceptions
`fcmpo` *DST*, *SRC1*, *SRC2*	*DST* ← (*SRC1*) compare (*SRC2*)	FPRs	CR field	`I - - - -`
`fcmpu` *DST*, *SRC1*, *SRC2*	*DST* ← (*SRC1*) compare (*SRC2*)	FPRs	CR field	- - - - -

Table F-11 Arithmetic instructions

Instruction	Description	SRC	DST	Exceptions
`fadd`* *DST*, *SRC1*, *SRC2*	*DST* ← (*SRC1*) + (*SRC2*)	FPRs	FPR	I X O U -
`fsub`* *DST*, *SRC1*, *SRC2*	*DST* ← (*SRC1*) – (*SRC2*)	FPRs	FPR	I X O U -
`fmul`* *DST*, *SRC1*, *SRC2*	*DST* ← (*SRC1*) × (*SRC2*)	FPRs	FPR	I X O U -
`fdiv`* *DST*, *SRC1*, *SRC2*	*DST* ← (*SRC1*) / (*SRC2*)	FPRs	FPR	I X O U D

Table F-12 Multiply-add instructions

Instruction	Description	SRC	DST	Exceptions
`fmadd`* *DST*, *SRC1*, *SRC2*, *SRC3*	*DST* ← (*SRC1*) × (*SRC2*) + (*SRC3*)	FPRs	FPR	I X O U -
`fmsub`* *DST*, *SRC1*, *SRC2*, *SRC3*	*DST* ← (*SRC1*) × (*SRC2*) – (*SRC3*)	FPRs	FPR	I X O U -
`fnmadd`* *DST*, *SRC1*, *SRC2*, *SRC3*	*DST* ← – ((*SRC1*) × (*SRC2*) + (*SRC3*))	FPRs	FPR	I X O U -
`fnmsub`* *DST*, *SRC1*, *SRC2*, *SRC3*	*DST* ← – ((*SRC1*) × (*SRC2*) – (*SRC3*))	FPRs	FPR	I X O U -

Table F-13 Move instructions

Instruction	Description	SRC	DST	Exceptions
`fabs`* *DST*, *SRC*	*DST* ← |(*SRC*)|	FPR	FPR	- - - - -
`fmr`* *DST*, *SRC*	*DST* ← (*SRC*)	FPR	FPR	- - - - -
`fneg`* *DST*, *SRC*	*DST* ← – (*SRC*)	FPR	FPR	- - - - -
`fnabs`* *DST*, *SRC*	*DST* ← – |(*SRC*)|	FPR	FPR	- - - - -