ROC24_minmin ISA details

James C Brakefield © 2024

Data formats, unsigned only for minmin project			
ISA has provision for four data types each of four lengths		open issues as to sign extension versus zero extension, and whether to have non-interleaved/bit-reversed float formats?	
unsigned 8	`nnnnnnnn`	little endian, ie bits numbered right to left zero to 7	displacements are 2's complement
unsigned 16	`nnnnnnnn nnnnnnnn`	little endian, ie bits numbered right to left zero to 15	displacements are 2's complement
unsigned 24	`nnnnnnnn nnnnnnnn nnnnnnnn`	little endian, ie bits numbered right to left zero to 23	displacements are 2's complement
unsigned 32	`nnnnnnnn nnnnnnnn nnnnnnnn nnnnnnnn`	little endian, ie bits numbered right to left zero to 31	displacements are 2's complement
sign and magnitude	`nnnnnnns etc`	sign and magnitude, code for minus zero used for NaN	lengthened by zero extension
2's complement	`snnnnnnn etc`	2's complement	lengthened by sign extension
float8	`mmmm.1eess`	8-bit float, signed exponent and signed mantissa	implied leading 1. mantissa bit, mantissa bit reversed, lengthened by zero extension
float16	`mmmmmmme mmmm.1eess`	16-bit float, signed exponent and signed mantissa	implied leading 1. mantissa bit, mantissa bit reversed, lengthened by zero extension
float24	`mmmmmmee mmmmmmme mmmm.1eess`	24-bit float, signed exponent and signed mantissa	implied leading 1. mantissa bit, mantissa bit reversed, lengthened by zero extension
float32	`mmmmmmee mmmmmmee mmmmmmme mmmm.1eess`	interleaved mantissa and exponent bits allow zero extension with exponent lengthening	
Posit		differs from float in that all but two exponent bits derived from trailing zero bits count of the mantissa	
Immediate values encoded within instruction stream			
immediate values	`1snnn nnnnnnn`	12-bit displacement	used for short branch displacements or value of -2048 to 2047
immediate values	`1nnnn nnllllll`	6-bit bit location and 6-bit length	used in shift and bit field insert/extract instructions
immediate values	`00nnn`	value of 0…7	short constant
immediate values	`011nn`	value of -4…-1	short constant
immediate values	`010nn`	one to four byte value follows instruction	one to four byte value
Special registers			
register zero used to hold return address			
last four registers are residue, frame pointer, stack pointer & PC		residue register gets the carry/overflow from add/subtract, the remainder from divide, upper half of multiply and rounding error from floating-point	
residue as base register & register zero as index register read as zero			
Type of operation determined by types of the operands, so same op-code for unsigned, signed, float or Posit; mismatched operand types may cause a trap			
each register has a 2+ bit type code: unsigned, signed, float & Posit(26+ bits total); additional bits for floating point exponent and Posit mantissa			

format name	instruction layout	usage	address arithmetic	
fmt24drsi	`sssss rrrrr i ddddd 00xxxxxx`	basic 24-bit instruction		x: op-code bit
fmt24drs	`sssss rrrrr 0 ddddd 00xxxxxx`	R op S ->D		i: immediate enable
fmt24dri	`nnnnn rrrrr 1 ddddd 00xxxxxx`	R op immediate ->D		m, n: -4..7 or byte length of following data or 12-bit signed
fmt24dbx	`jjjjj bbbbb 0 ddddd 00xxxxds`	load/store base + scaled index	B + J * ds	ds: data size 1..4, derived from op-code LSB
fmt24dbn	`nnnnn bbbbb 1 ddddd 00xxxxxx`	load/store base + offset	B + N	
fmt24ds	`sssss xxxxx z ddddd 00xxxxxx`	op R -> D; 32 single operand instructions		z: sign change or 1's complement
fmt24dn	`nnnnn nnnnn n ddddd 00xxxxxx`	eleven bit signed offset or displacement	PC + N	
fmt24dcn	`nnnnn nnn ccc ddddd 00xxxxxx`	conditional relative branch	PC + N	c: =0,>0,>=0,even & their reverse; for floats: sign, denorm, exact &
fmt24dan	`nnnnn nnnaa x ddddd 00xxxxxx`	eight bit immediate with type code		a: data type unsigned, signed, float, posit

format name	instruction layout	usage	address arithmetic	
fmt32drsc	`ttttt zzu sssss rrrrr i ddddd 01xxxxxx`	basic 32-bit instruction		u: residue update enable
fmt32dbjs	`jjjjj yyu sssss bbbbb i ddddd 01xxxxxx`	load from base + index * scaling + offset	B + J * Y * ds + S	y: 1..4
fmt32dbjn	`jjjjj yyu nnnnn bbbbb 1 ddddd 01xxxxxx`	load from base + index * scaling + offset	B + J * Y * ds + N	v: negate J
fmt32dbjs	`jjjjj yy0 sssss bbbbb 0 ddddd 01xxxxxx`	store to base + index * scaling + offset	B + J * Y * ds + S	v: swap operands enable
fmt32dbjn	`jjjjj yy0 nnnnn bbbbb 1 ddddd 01xxxxxx`	store to base + scaled index + offset	B + J * Y * ds + N	v: add one to J * Y * ds
fmt32mbjs	`jjjjj yy1 sssss bbbbb 0 mmmmm 01xxxxxx`	store immediate to base + index * scaling + offset	B + J * Y * ds + S	
fmt32mbjn	`jjjjj yy1 nnnnn bbbbb 1 mmmmm 01xxxxxx`	store immediate to base + index * scaling + offset	B + J * Y * ds + N	two immediates in one instruction?
fmt32ds	`llzaa xxu sssss xxxxx i ddddd 01xxxxxx`	op R -> D; 128 single operand instructions		length, sign flip, type, residue update, immediate
fmt32drst	`ttttt zzu sssss rrrrr i ddddd 01xxxxxx`	op(R, S, T) -> D; triple operand instructions		sign flip for R & T, residue update enable
fmt32dn	`nnnnn nnn nnnnn nnnnn n ddddd 01xxxxxx`	nineteen bit signed offset or immediate	PC + N	

fmt32dan		nnnnn nnn nnnnn nnnaa x ddddd 01xxxxxx			sixteen bit immediate with type code			
fmt32dcn		nnnnn nnn nnnnn nnn ccc ddddd 01xxxxxx			conditional relative branch		PC + N	c: =0,>0,>=0,even & their reverse; for floats: sign, denorm, exact &
fmt40drsct		ttttt xxx ccccc xxu sssss rrrrr i ddddd 10xxxxxx				basic 40-bit instruction w	basic 40-bit instruction with three source registers & predication	
fmt40dan		nnnnnnnn nnnnnnnn nnnnnnnn aa x ddddd 10xxxxxx				40-bit instruction to load 24-bit typed value		
fmt48dan		nnnnnnnn nnnnnnnn nnnnnnnn nnnnnnnn aa x ddddd 11xxxxxx				48-bit instruction to load 32-bit typed value		

format name	N	op-code bits			neumonic	description		comments
Generic ADD/SUB			32-bit format has options for residue update and R & T sign change			R + S -> D or R + N -> D		**mixed data type operations may result in a trap**
fmt24drs, fmt24drn	5+	00100000			ADD, ADDI, FADD, FADDI ...	R + S -> D		
fmt24drs, fmt24drn	5+	00100100			ADDNU, ADDNUI	R + S -> D		no residue register update
fmt32drst, fmt32drnt	5+	01100000			ADD3, ADD3I	R + S + T -> D		
fmt24drs, fmt24drn	5+	00100001			SUB, SUBI	R - S -> D		
fmt24drs, fmt24drn	5+	00100101			SUBNU, SUBNUI	R - S -> D		no residue register update
fmt32drst, fmt32drnt	5+	01100001			SUB3, SUB3I	R - S + T -> D		
fmt24drs, fmt24drn	5+	00101011			CMP, CMPI	R - S comparison result to D register		My 66000 CCR format plus even/odd bits
Generic Boolean			32-bit format has options for residue update and R & T 1's complement					data types are ignored
fmt24drs, fmt24drn	5+	00101000			AND, ANDI	R and S -> D		data types are ignored
fmt24drs, fmt24drn	5+	00101001			OR, ORI	R or S -> D		data types are ignored
fmt24drs, fmt24drn	5+	00101010			XOR, XORI	R xor S -> D		data types are ignored
fmt32drsc, fmt32drnc	5+	01101000			AND3, AND3I	R and S and T -> D		data types are ignored
fmt32drsc, fmt32drnc	5+	01101001			OR3, OR3I	R or S or T -> D		data types are ignored
fmt32drsc, fmt32drnc	5+	01101010			XOR3, XOR3I	R xor S xor T -> D		data types are ignored
Generic MUL/DIV			32-bit format has options for residue update and R & T sign change					**mixed data type operations may result in a trap**
fmt24drs, fmt24drn	5+	00100010			MUL, MULI	R * S -> D		residue register update, 32-bit version has residue enable bit
fmt24drs, fmt24drn	5+	00100110			MULNU, MULNUI	R * S -> D		no residue register update
fmt32drsc, fmt32drnc	5+	01100010			MAC, MACI	R * S + T -> D		multiply and add
fmt24drs, fmt24drn	5+	00100011			DIV, DIVI	R / S -> D	DDIV	residue register update, 32-bit version has residue enable bit
fmt24drs, fmt24drn	5+	00100111			DIVNU, DIVNUI	R / S -> D		no residue register update
fmt32drsc, fmt32drnc	5+	01100011			DDIV, DDIVI	R&T / S -> D		divide with double length dividend
Generic shift/extract/insert								
fmt24drs, fmt24drn	5+	00101110			ASR, ASRI			arithmetic shift right, signed shift count
fmt24drs, fmt24drn	5+	00101101			LSR, LSRI			logical shift right, signed shift count
fmt24drs, fmt24drn	5+	00101111			ROR, RORI			rotate right, signed shift count
fmt32drsc, fmt32drnc	5+	01101110			ASR, ASRI			double length arithmetic shift right, signed shift count
fmt32drsc, fmt32drnc	5+	01101101			SHR, SHRI			double length logical shift right, signed shift count
fmt32drsc, fmt32drnc	5+	01101111			ROR, RORI			double length rotate right, signed shift count
fmt24drs, fmt24drn	5+	00101100			EXTRCT, EXTRCTI	extract bit field	starting bit and length concatenated into S or n as a 12-bit contiguous field	
fmt32drsc, fmt32drnc	5+	01101100			EXTRCT, EXTRCTI	extract bit field		double length source
fmt32drsc, fmt32drnc	5+	01101011			INSRT, INSRTI	bit field insert	starting bit and length concatenated into S or N as a 12-bit contiguous field	
fmt32drst, fmt32drnt	5+	01110110			MERG			Select bits from two operands
fmt32drst, fmt32drnt	5+	01110101			CMOV			Select one of two operands
Load/Store			32-bit format has additional scaling of index register					
fmt24dbj, fmt24dbn	5+	000110xx			FLD8, FLD16, FLD24, FLD32	floating-point load		separate floating-point load inst allow a simpler data format
fmt24dbj, fmt24dbn	5+	000101xx			LD8, LD16, LD24, LD32	load signed byte/half/word from memory		
fmt24dan	8	00001000			LDI8, LDI16, LDI24, LDI32	load typed immediate		can be used as absolute jump if PC is R31
fmt24dbj, fmt24dbn	5+	000100xx			LDU8, LDU16, LDU24, LDU32	load unsigned byte/half/word from memory		
fmt32dbjs, fmt32dbjn	5+	011001xx			LEA8, LEA16, LEA24, LEA32	load effective address		can be used as absolute jump if PC is R31
fmt24dbj, fmt24dbn	5+	000111xx			PLD8, PLD16, PLD24, PLD32	Posit load		separate floating-point load inst allow a simpler data format
fmt24dbj, fmt24dbn	5+	000011xx			ST8, ST16, ST24, ST32	store byte/half/word to memory		32-bit version supports store immediate
fmt32dbjs, fmt32dbjn	5+	010110xx			FLD8, FLD16, FLD24, FLD32	floating-point load		separate floating-point load inst allow a simpler data format
fmt32dbjs, fmt32dbjn	5+	010101xx			LD8, LD16, LD24, LD32	load signed byte/half/word from memory		

fmt32dan	16	01001000			LDI8, LDI16, LDI24, LDI32	load typed immediate		can be used as absolute jump if PC is R31
fmt32dbjs, fmt32dbjn	5+	010100xx			LDU8, LDU16, LDU24, LDU32	load unsigned byte/half/word from memory		
fmt32dbjs, fmt32dbjn	5+	010111xx			PLD8, PLD16, PLD24, PLD32	Posit load		separate floating-point load inst allow a simpler data format
fmt32dbjs, fmt32dbjn	5+	010011xx			ST8, ST16, ST24, ST32	store byte/half/word to memory		32-bit version supports store immediate
Control								
fmt8		00000000			TRAP	illeagal instruction trap		one byte inst
fmt8		00000001			BRK	software breakpoint		one byte inst
fmt8		00000010			NXTH	force two byte alignment		one byte inst
fmt8		00000011			NXTW	force four byte alignment		one byte inst
fmt24drn	5+	00000110			BBS, BBC	relative branch on bit set/clear		immediate option bit is branch on set/clear
fmt24n	16	00000100			BR	branch relative		16-bit displacement
fmt32n	24	01000100			BR			24-bit displacement
fmt24dcn	8	00000111			BRcc	conditional relative branch/jump on regist		eight condition codes: =0,>0,>=0,even and the reverse
fmt32dcn	16	01000111			BRcc			
fmt24dbj, fmt24dbn	5+	00110110			CALL	LEA to PC		return address to any register
fmt32dbjs, fmt32dbjn	5+	01110110			CALL	LEA to PC		
fmt24n	16	00000101			CALLR	call relative		16-bit displacement, return address to R0
fmt32n	24	01000101			CALLR	call relative		24-bit displacement, return address to R0
					ENTER	create frame and save		My 66000 instruction
					EXIT	drop frame and restore registers		My 66000 instruction
fmt24drs, fmt24drn		00110011	TBD	01111001	LOOP, LOOPI	vector loop instruction		My 66000 instruction
fmt24dn		00110010	fmt32dn	01111000	VECT	identify loop vector registers		My 66000 instruction
fmt32		01111100			PCB1	create predicate shadow on bit		My 66000 instruction
fmt32		01111011			PCND	create predicate shadow on condition		My 66000 instruction
Irregular								
fmt24ds		00110111			single operand instructions		negate/invert supported	
fmt32ds		01110111			single operand instructions		length, sign flip, type, residue update, immediate	
fmt24drs, fmt24drn					EADD, EADDI	add to exponent		
fmt24drs, fmt24drn		00110100			MAX, MAXI	maximum		
fmt32drsc, fmt32drnc	5+	01110000			MAX3, MAX3I	maximum		TBD residue result
fmt32drsc, fmt32drnc	5+	01110010			MEDIAN3	median		TBD residue result
fmt24drs, fmt24drn		00110101			MIN, MINI	minimum		
fmt32drsc, fmt32drnc	5+	01110001			MIN3, MIN3I	minimum		TBD residue result
fmt32drst, fmt32drnt	5+	01110011			INTP-BLEND-LERP, FINTP	linear interpolation		
fmt32drst, fmt32drnt		01111110			LDM	multi-register load from r	My 66000 inst.	28-bit store (includes register type code)
fmt32drst, fmt32drnt	5+	01110100			LUT	Boolean predicate over multiple bits (5 to 8), R & T are start and ending register numbers which sets the size o		
fmt32		01111010			MMOV	memory to memory byte	My 66000 instruction	
fmt32					MV2	move two registers		can be used to swap registers
fmt32drsc, fmt32drnc					SHL, SHLI	shift left		
fmt32drst, fmt32drnt		01111101			STM	multi-register save to me	My 66000 inst.	28-bit store (includes register type code)
fmt32		01111111			TT	jump table dispatch	My 66000 instruction	

Single Operand Instructions (30 allocated of 32 in fmt24, or 56 of 128 in fmt32), operand negate/invert supported, 32-bit versions can set result length									
fmt24ds 00110111			fmt32ds 01110111			IN	read input port		S or n is the port number
fmt24ds			fmt32ds			OUT	write to output port		S or n is the port number
fmt24ds			fmt32ds			MOV, MOVI	move register	macro op	
fmt24ds			fmt32ds			NOT	Boolean complement	macro op	
fmt24ds			fmt32ds			NEG	Negate	macro op	

fmt24ds			fmt32ds			INC, DEC	Increment, Decrement	macro op	
fmt24ds			fmt32ds			ABS, NABS	Absolute value	macro op	
fmt24ds			fmt32ds			LDZCNT, LD1CNT, TRZCNT, TR1CNT	Leading and trailing zero/	result ranges from zero to register size	
fmt24ds			fmt32ds			POPCNT	Count of one/zero bits	result ranges from zero to register size	
fmt24ds			fmt32ds			SINPI, COSPI, TANPI	Trig functions in rotations		
fmt24ds			fmt32ds			ASINPI, ACOSPI, ATANPI	Trig functions in rotations		
fmt24ds			fmt32ds			INT, FLT	Integer from/to float		
			fmt32ds			CVT (48)	to & from: single, double	My 66000 instruction	
fmt24ds			fmt32ds			EXPON	extract exponent		
fmt24ds			fmt32ds			FRACT	extract fraction		
fmt24ds			fmt32ds			SQRT	integer square root		
fmt24ds			fmt32ds			FSQRT	floating-point square root		
fmt24ds			fmt32ds			FRSQRT	floating-point reciprical square root		
fmt24ds			fmt32ds			FRCP	reciprical	macro op	
fmt24ds			fmt32ds			LN2P1	log to base 2 plus one		
fmt24ds			fmt32ds			LN2	log to base 2		
			fmt32ds			LNP1	natural log plus one		
			fmt32ds			LN	natural log		
			fmt32ds			LOGP1	base 10 logarithm plus one		
			fmt32ds			LOG	base 10 logarithm		
fmt24ds			fmt32ds			EXP2M1	base 2 exponentation minus one		
fmt24ds			fmt32ds			EXP2	base 2 exponentation		
			fmt32ds			EXPM1	base E exponentation minus one		
			fmt32ds			EXP	base E exponentation		
			fmt32ds			EXP10M1	base 10 exponentation minus one		
			fmt32ds			EXP10	base 10 exponentation		
			fmt32ds			SIN	trig functions in radians		
			fmt32ds			COS	trig functions in radians		
			fmt32ds			TAN	trig functions in radians		
			fmt32ds			ASIN	trig functions in radians		
			fmt32ds			ACOS	trig functions in radians		
			fmt32ds			ATAN	trig functions in radians		
fmt24ds			fmt32ds			RND	round using current mode		
fmt24ds			fmt32ds			RNDTEV	round to nearest, ties to even		
fmt24ds			fmt32ds			RNDTOD	inexact round to nearest half odd		
fmt24ds			fmt32ds			RNDTZ	round toward zero		
fmt24ds			fmt32ds			RNDFZ	round away from zero		
fmt24ds			fmt32ds			CEIL	round up		
fmt24ds			fmt32ds			FLOOR	round down		
							statistical rounding		
									Totals