ROC24_3sz operator list

James C Brakefield © 2023

register zero used to hold return address	"n" immediate field is a value of -4..7, first 4-bits of 12-bit immediate, or length of postfixed immediate in bytes		
data sizes: byte(8), half(16), word(24)	last four registers are residue, frame pointer, stack pointer & PC		
Y scale factor is data size times (1 to 4)	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 4-bit type code: unsigned, signed, float & Posit(28-bits total)	32-bit instructions: residue update optional		

format name	instruction layout	usage		address arithmetic
fmt24drsi	`sssss rrrrr i ddddd 00xxxxxx`	basic 24-bit instruction		
fmt24drs	`sssss rrrrr 0 ddddd 00xxxxxx`	R op S ->D		
fmt24dri	`nnnnn rrrrr 1 ddddd 00xxxxxx`	R op immediate ->D		
fmt24dbx	`jjjjj bbbbb 0 ddddd 00xxxxxx`	load/store base + scaled index		B + J * ds
fmt24dbn	`nnnnn bbbbb 1 ddddd 00xxxxxx`	load/store base + offset		B + N
fmt24dr	`xxxxx rrrrr z ddddd 00xxxxxx`	op R -> D; 32 single operand instructions		
fmt24dn	`nnnnn nnnnn n ddddd 00xxxxxx`	eleven bit signed offset or displacement		PC + N
fmt24dan	`nnnnn nnnaa x ddddd 00xxxxxx`	eight bit immediate with type code		

format name	instruction layout	usage		address arithmetic
fmt32drsc	`ttttt zzu sssss rrrrr i ddddd 01xxxxxx`	basic 32-bit instruction		
fmt32drsc	`ccccc zzu sssss rrrrr i ddddd 01xxxxxx`	R op S ->D; execute on predicate match		
fmt32drnc	`ccccc zzu nnnnn rrrrr 1 ddddd 01xxxxxx`	R op imm ->D; execute on predicate match		
fmt32dbjs	`jjjjj yyu sssss bbbbb i ddddd 01xxxxxx`	load from base + index * scaling + offset		B + J * Y * ds + S
fmt32dbjn	`jjjjj yyu nnnnn bbbbb 1 ddddd 01xxxxxx`	load from base + index * scaling + offset		B + J * Y * ds + N
fmt32dbjs	`jjjjj yy0 sssss bbbbb 0 ddddd 01xxxxxx`	store to base + index * scaling + offset		B + J * Y * ds + S
fmt32dbjn	`jjjjj yy0 nnnnn bbbbb 1 ddddd 01xxxxxx`	store to base + scaled index + offset		B + J * Y * ds + N
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
fmt32dr	`ccccc xxu xxxxx rrrrr z ddddd 01xxxxxx`	op R -> D; 128 single operand instructions		
fmt32drst	`ttttt zzu sssss rrrrr i ddddd 01xxxxxx`	op(R, S, T) -> D; triple operand instructions		
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		
fmt40drsct	`ttttt xxx ccccc xxu sssss rrrrr i ddddd 10xxxxxx`	basic 40-bit instruction with three source registers & predication		
fmt40dan	`nnnnn nnn nnnnn nnn nnnnnnnn aa x ddddd 10xxxxxx`	40-bit instruction to load 24-bit typed value		

24-bit formats	32-bit formats	neumonic	short description (44 of 64 allocated)	comments	
	Prunned instructions in grey, e.g. need to use 40-bit instruction with its additional op-code bits				# op codes
fmt24dbj, fmt24dbn	fmt32dbjs, fmt32dbjn	LDU8, LDU16, LDU24	load unsigned byte/half/word from memory		
fmt24dbj, fmt24dbn	fmt32dbjs, fmt32dbjn	LD8, LD16, LD24	load signed byte/half/word from memory		
fmt24dbj, fmt24dbn	fmt32dbjs, fmt32dbjn	FLD8, FLD16, FLD24	floating-point load		
fmt24dbj, fmt24dbn	fmt32dbjs, fmt32dbjn	PLD8, PLD16, PLD24	Posit load		

fmt24dan	fmt32dan	LDI8, LDI16, LDI24	load typed immediate		
	fmt32dbjs, fmt32dbjn	LEA8, LEA16, LEA24	load effective address		
fmt24dbj, fmt24dbn	fmt32dbjs, fmt32dbjn	ST8, ST16, ST24	store byte/half/word to memory	32-bit version supports store immediate	
fmt24drs, fmt24drn	fmt32drst, fmt32drnt	ADD, ADDI	R + S -> D		
fmt24drs, fmt24drn	fmt32drst, fmt32drnt	SUB, SUBI	R - S -> D		
fmt24drs, fmt24drn	fmt32drsc, fmt32drnc	MUL, MULI	R * S -> D		
	fmt32drst, fmt32drnt	MAC	multiply-add		
fmt24drs, fmt24drn	fmt32drsc, fmt32drnc	DIV, DIVI	R / S -> D		
fmt24drs, fmt24drn	fmt32drsc, fmt32drnc	AND, ANDI	R and S -> D	type code ignored	
fmt24drs, fmt24drn	fmt32drsc, fmt32drnc	OR, ORI	R or S -> D	type code ignored	
fmt24drs, fmt24drn	fmt32drsc, fmt32drnc	XOR, XORI	R xor S -> D	type code ignored	
fmt24drs, fmt24drn	fmt32drsc, fmt32drnc	CMP, CMPI	R : S -> D, result to D register	My 66000 CCR format plus even/odd bits	
fmt24drs, fmt24drn	fmt32drsc, fmt32drnc	FATAN2PI	arc-tangent in rotations for two operands		
fmt24drs, fmt24drn	fmt32drsc, fmt32drnc	FPOW, FPOWI	R raised to the S power		
	fmt32drsc, fmt32drnc	INSRT, INSRTI	bit field insert	starting bit and length concatenated into S or N as a 12-bi	
fmt24drs, fmt24drn	fmt32drsc, fmt32drnc	EXTRCT, EXTRCTI	extract bit field	starting bit and length concatenated into S or N as a 12-bi	
fmt24drs, fmt24drn	fmt32drsc, fmt32drnc	ROL, ROLI	rotate left		
fmt24drs, fmt24drn	fmt32drsc, fmt32drnc	SHR, SHRI	shift right		
fmt24drs, fmt24drn	fmt32drsc, fmt32drnc	ASR, ASRI	arithmetic shift right		
fmt24drs, fmt24drn	fmt32drsc, fmt32drnc	SHL, SHLI	shift left		
fmt24cbj, fmt24cbn	fmt32cbjs, fmt32cbjn	JMPcc, CALLcc	conditional jump/call	D is the condition, JMP-never mapped to CALL	
fmt24cn	fmt32cn	BRcc, BSRcc	conditional relative branch/call with 1	D is the condition, BR-never mapped to CALL	
fmt24drs, fmt24drn	TBD	BBS, BBC	relative branch on bit set/clear	N: branch delta, R: source, D: bit number; requires 4 op-co	
fmt24crn	fmt22crbj, fmt22crbn	BRccRC, JMPccRC	conditional relative branch/jump on register contents		
fmt24drs, fmt24drn	fmt32drsc, fmt32drnc	MAX, MAXI	maximum		
fmt24drs, fmt24drn	fmt32drsc, fmt32drnc	MIN, MINI	minimum		
	fmt32drsc, fmt32drnc	MEDIAN	median		
fmt24drs, fmt24drn	fmt32drsc, fmt32drnc	EADD, EADDI	add to exponent		
fmt24dn	fmt32dn	VECT	identify loop vector registers	My 66000 instruction	
fmt24drs, fmt24drn	TBD	LOOP, LOOPI	vector loop instruction	My 66000 instruction	
fmt24dr	fmt32dr		single operand instructions	operand can be negated/inverted	
		Probably useful instructions			
fmt24drs, fmt24drn		MMOV	memory to memory byte string move	My 66000 instruction	
	TBD	PCND	create predicate shadow on conditior	My 66000 instruction	
	TBD	PCB1	create predicate shadow on bit	My 66000 instruction	
TBD	TBD	STM	multi-register save to memory	My 66000 inst.	28-bit store
TBD	TBD	LDM	multi-register load from memory	My 66000 inst.	28-bit load
					Totals

Single Operand Instructions (30 allocated of 32 in fmt24, or 92 of 128 in fmt32), operand negate/invert supported		

	Prunned 24-bit instructions in grey, e.g. need to use 32-bit instruction with its additional op-code bits				
fmt24dr	fmt32dr	IN	read input port (z=0)	R is the port number	
fmt24dr	fmt32dr	OUT	write to output port (z=1)	R is the port number	
fmt24dr	fmt32dr	MOV, MOVI	move register	macro op	
fmt24dr	fmt32dr	NOT	Boolean complement	macro op	
fmt24dr	fmt32dr	NEG	Negate	macro op	
fmt24dr	fmt32dr	INC, DEC	Increment, Decrement	macro op	
fmt24dr	fmt32dr	ABS, NABS	Absolute value	macro op	
fmt24dr	fmt32dr	LDZCNT, LD1CNT, TRZCNT, TR1CNT	Leading and trailing zero/ones count	ranges from zero to register size	
fmt24dr	fmt32dr	POPCNT	Count of one/zero bits	ranges from zero to register size	
fmt24dr	fmt32dr	SINPI, COSPI, TANPI	Trig functions in half rotations		
fmt24dr	fmt32dr	ASINPI, ACOSPI, ATANPI	Trig functions in half rotations		
fmt24dr	fmt32dr	INT, FLT	Integer from/to float		
	fmt32dr	CVT (48)	to & from: single, double, unsigned, si	My 66000 instruction	
fmt24dr	fmt32dr	EXPON	extract exponent		
fmt24dr	fmt32dr	FRACT	extract fraction		
fmt24dr	fmt32dr	SQRT	square root		
fmt24dr	fmt32dr	FSQRT	floating-point square root		
fmt24dr	fmt32dr	FRSQRT	floating-point reciprical square root		
fmt24dr	fmt32dr	FRCP	reciprical	macro op	
fmt24dr	fmt32dr	LN2P1	log to base 2 plus one		
fmt24dr	fmt32dr	LN2	log to base 2		
	fmt32dr	LNP1	natural log plus one		
	fmt32dr	LN	natural log		
	fmt32dr	LOGP1	base 10 logarithm plus one		
	fmt32dr	LOG	base 10 logarithm		
fmt24dr	fmt32dr	EXP2M1	base 2 exponentation minus one		
fmt24dr	fmt32dr	EXP2	base 2 exponentation		
	fmt32dr	EXPM1	base E exponentation minus one		
	fmt32dr	EXP	base E exponentation		
	fmt32dr	EXP10M1	base 10 exponentation minus one		
	fmt32dr	EXP10	base 10 exponentation		
	fmt32dr	SIN	trig functions in radians		
	fmt32dr	COS	trig functions in radians		
	fmt32dr	TAN	trig functions in radians		
	fmt32dr	ASIN	trig functions in radians		
	fmt32dr	ACOS	trig functions in radians		
	fmt32dr	ATAN	trig functions in radians		
fmt24dr	fmt32dr	RND	round using current mode		
fmt24dr	fmt32dr	RNDTEV	round to nearest, ties to even		
fmt24dr	fmt32dr	RNDTOD	inexact round to nearest odd		

fmt24dr	fmt32dr	RNDTZ	round toward zero		
fmt24dr	fmt32dr	RNDFZ	round away from zero		
fmt24dr	fmt32dr	CEIL	ceiling		
fmt24dr	fmt32dr	FLOOR	floor		
					Totals

Triple Operand Instructions, typically 32 or 40-bit instructions, see also RTF64 by Robert Finch					
	fmt32drst, fmt32drnt	INTP-BLEND-LERP, FINTP	linear interpolation		
	fmt32drst, fmt32drnt	LUT	Boolean predicate over multiple bits (5 to 8), R & T are start and ending register numbers which		
	fmt32drst, fmt32drnt	CMOV	Select one of two operands		
	fmt32drst, fmt32drnt	MERG	Select bits from two operands		