ROC32_8 operator list

James C Brakefield © 2020

register zero used to hold return address
data sizes: byte(8), half(16), word(32), long(64)
Y scale factor is data size times (1 to 4)
combined interger & floating-point register file

immediate field is value of -8..15 or length of postfixed immediate in bytes
last four registers are condition code, frame pointer, stack pointer & PC
CCR as base register & register zero as index register read as zero

format name	instruction layout	usage	address arithmetic
fmt24drsi	00xxxxxx ddddd i rrrrr sssss	basic 24-bit instruction	
fmt24drs	00xxxxxx ddddd 0 rrrrr sssss	R op S ->D	
fmt24drn	00xxxxxx ddddd 1 rrrrr nnnnn	R op immediate ->D	
fmt24dbj	00xxxxxx ddddd 0 bbbbb jjjjj	load/store base + scaled index	B + J * ds
fmt24dbn	00xxxxxx ddddd 1 bbbbb nnnnn	load/store base + offset	B + N
fmt24dr	00xxxxxx ddddd z rrrrr xxxxx	op R -> D; 32 single operand instructions	
fmt24dn	00xxxxxx ddddd n nnnnn nnnnn	eleven bit signed offset or immediate	PC + N

format name	instruction layout	usage	address arithmetic
fmt32drsc	01xxxxxx ddddd i rrrrr sssss zzu ccccc	basic 32-bit instruction	
fmt32drsc	01xxxxxx ddddd 0 rrrrr sssss zzu ccccc	R op S ->D; execute on predicate match	
fmt32drnc	01xxxxxx ddddd 1 rrrrr nnnnn zvu ccccc	R op imm ->D; execute on predicate match	
fmt32dbjs	01xxxxxx ddddd 0 bbbbb sssss yyu jjjjj	load from base + index * scaling + offset	B + J * Y + S
fmt32dbjn	01xxxxxx ddddd 1 bbbbb nnnnn yyu jjjjj	load from base + index * scaling + offset	B + J * Y + N
fmt32dbjs	01xxxxxx ddddd 0 bbbbb sssss yy0 jjjjj	store to base + index * scaling + offset	B + J * Y + S
fmt32dbjn	01xxxxxx ddddd 1 bbbbb nnnnn yy0 jjjjj	store to base + index * scaling + offset	B + J * Y + N
fmt32mbjs	01xxxxxx mmmmm 0 bbbbb sssss yy1 jjjjj	store immediate to base + index * scaling + offset	B + J * Y + S
fmt32mbjn	01xxxxxx mmmmm 1 bbbbb nnnnn yy1 jjjjj	store immediate to base + index * scaling + offset	B + J * Y + N
fmt32drst	01xxxxxx ddddd 0 rrrrr sssss zzu ttttt	op(R, S, T) -> D; triple operand instructions	
fmt32drnt	01xxxxxx ddddd 1 rrrrr nnnnn zzu ttttt	op(R, N, T) -> D; triple operand instructions	
fmt32dr	00xxxxxx ddddd z rrrrr xxxxx xxu ccccc	op (R) -> D; 128 single operand instructions	
fmt32dn	01xxxxxx ddddd n nnnnn nnnnn nnn nnnnn	nineteen bit signed offset or immediate	PC + N
fmt40drsct	10xxxxxx ddddd i rrrrr sssss zzu ccccc xxx ttttt	basic 40-bit instruction with three source registers & predi	
fmt48drscte	11xxxxxx ddddd i rrrrr sssss zzu ccccc xxx ttttt xxx eeeee	basic 48-bit instruction with 3 source, 2 destination & pred	
fmt48drsctwe	11xxxxxx ddddd i rrrrr sssss zzu ccccc xww ttttt www eeeee	basic 48-bit instruction with 4 source, 2 destination & pred	

24-bit formats	32-bit formats	neumonic	short description (72 of 64 allocated)	comments
fmt24dbj, fmt24dbn	fmt32dbjs, fmt32dbjn	LDU8	load unsigned byte from memory	
fmt24dbj, fmt24dbn	fmt32dbjs, fmt32dbjn	LDU16	load unsigned half from memory	
fmt24dbj, fmt24dbn	fmt32dbjs, fmt32dbjn	LDU32	load unsigned word from memory	

fmt24dbj, fmt24dbn	fmt32dbjs, fmt32dbjn	LD64	load unsigned long from memory	
fmt24dbj, fmt24dbn	fmt32dbjs, fmt32dbjn	LDI8	load signed byte from memory	
fmt24dbj, fmt24dbn	fmt32dbjs, fmt32dbjn	LDI16	load signed half from memory	
fmt24dbj, fmt24dbn	fmt32dbjs, fmt32dbjn	LDI32	load signed word from memory	
fmt24dbj, fmt24dbn	fmt32dbjs, fmt32dbjn	ST8	store byte to memory	32-bit version supports store immediate
fmt24dbj, fmt24dbn	fmt32dbjs, fmt32dbjn	ST16	store half to memory	32-bit version supports store immediate
fmt24dbj, fmt24dbn	fmt32dbjs, fmt32dbjn	ST32	store word to memory	32-bit version supports store immediate
fmt24dbj, fmt24dbn	fmt32dbjs, fmt32dbjn	ST64	store long to memory	32-bit version supports store immediate
fmt24drs, fmt24drn	fmt32drsc, fmt32drnc	ADD, ADDI	R + S -> D	contents of register R + contents of register S to register D
fmt24drs, fmt24drn	fmt32drsc, fmt32drnc	SUB, SUBI	R - S -> D	
fmt24drs, fmt24drn	fmt32drsc, fmt32drnc	ADC, ADCI	R + S + carry -> D	
fmt24drs, fmt24drn	fmt32drsc, fmt32drnc	SBC, SBCI	R - S + carry -> D	
fmt24drs, fmt24drn	fmt32drsc, fmt32drnc	MUL, MULI	R * S -> D	
fmt24drs, fmt24drn	fmt32drsc, fmt32drnc	MULU, MULUI	unsigned multiply, upper half result	
fmt24drs, fmt24drn	fmt32drsc, fmt32drnc	MULUS, MULUSI	signed multiply, upper half result	
fmt24drs, fmt24drn	fmt32drsc, fmt32drnc	DIVU, DIVUI	R / S -> D, unsigned	
fmt24drs, fmt24drn	fmt32drsc, fmt32drnc	DIVS, DIVSI	signed integer divide	
fmt24drs, fmt24drn	fmt32drsc, fmt32drnc	REMU, REMUI	unsigned integer divide remainder	
fmt24drs, fmt24drn	fmt32drsc, fmt32drnc	REMS, REMSI	signed integer divide remainder	
fmt24drs, fmt24drn	fmt32drsc, fmt32drnc	AND, ANDI	R and S -> D	
fmt24drs, fmt24drn	fmt32drsc, fmt32drnc	OR, ORI	R or S -> D	
fmt24drs, fmt24drn	fmt32drsc, fmt32drnc	XOR, XORI	R xor S -> D	
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	FADD, FADDI	floating-point add	
fmt24drs, fmt24drn	fmt32drsc, fmt32drnc	FSUB, FSUBI	floating-point subtract	
fmt24drs, fmt24drn	fmt32drsc, fmt32drnc	FMUL, FMULI	floating-point multiply	
	fmt32drst, fmt32drnt	FMA, FMAI	floating-point multiply and add	three operands
fmt24drs, fmt24drn	fmt32drsc, fmt32drnc	FDIV, FDIVI	floating-point divide	
fmt24drs, fmt24drn	fmt32drsc, fmt32drnc	FCMP, FCMPI	floating-point compare, result to D reg	My 66000 CCR format
fmt24drs, fmt24drn	fmt32drsc, fmt32drnc	FATAN2PI	arc-tangent in rotations for two operands	
fmt24drs, fmt24drn	fmt32drsc, fmt32drnc	FPOW, FPOWI	R raised to the S power	
fmt24drs, fmt24drn	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	unsigned extract bit field	starting bit and length concatenated into S or N as a 12-bi
fmt24drs, fmt24drn	fmt32drsc, fmt32drnc	EXTRCTS, EXTRCTSI	sign extended bit field extract	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	

fmt24dbj, fmt24dbn	fmt32dbjs, fmt32dbjn	JMPcc, CALLcc	conditional jump/call	D is the condition, JMP-never mapped to CALL
fmt24dn	fmt32dn	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 2/4 op-
fmt24drs, fmt24drn	fmt32drsc, fmt32drnc	MAX, MAXI	signed maximum	
fmt24drs, fmt24drn	fmt32drsc, fmt32drnc	MIN, MINI	signed minimum	
fmt24drs, fmt24drn	fmt32drsc, fmt32drnc	EADD, EADDI	add to exponent	
fmt24dn	fmt32dn	VECT	identify loop vector registers	
fmt24drs, fmt24drn	TBD	LOOP, LOOPI	vector loop instruction	
fmt24dn	fmt32dn	LDI	load immediate	
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 condition	My 66000 instruction
	TBD	PCB1	create predicate shadow on bit	My 66000 instruction
TBD	TBD	STM	multi-register save to memory	My 66000 instruction
TBD	TBD	LDM	multi-register load from memory	My 66000 instruction
fmt24dbj, fmt24dbn	fmt32drsc, fmt32drnc	FLD8, FLD16, FLD32, FLD64	floating-point load	could support 2nd 16-bit float format instead of the 8-bit
fmt24dbj, fmt24dbn	fmt32drsc, fmt32drnc	FST8, FST16, FST32, FST64	floating-point store	could support 2nd 16-bit float format instead of the 8-bit
fmt24drs, fmt24drn	fmt32drsc, fmt32drnc	FMAX, FMAXI	floating-point maximum	My 66000 instruction
fmt24drs, fmt24drn	fmt32drsc, fmt32drnc	FMIN, FMINI	floating-point minimum	My 66000 instruction
	fmt32dbjs, fmt32dbjn	LEA8, LEA16, LEA32, LEA64	load effective address	
Single Operand Instructions (92 allocated of 32 in fmt24, or of 128 in fmt32), operand negate/invert supported				
fmt24dr	fmt32dr	IN	read input port	R is the port number
fmt24dr	fmt32dr	OUT	write to output port	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, FABS, NABS, FNABS	Absolute value	macro op
fmt24dr	fmt32dr	LDZCNT, LD1CNT, TRZCNT, TR	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	
fmt24dr	fmt32dr	ANDCCR	clear bits in CCR	macro op
fmt24dr	fmt32dr	ORCCR	set bits in CCR	macro op

fmt24dr	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	
fmt24dr	fmt32dr	LNP1	natural log plus one	
fmt24dr	fmt32dr	LN	natural log	
fmt24dr	fmt32dr	LOGP1	base 10 logarithm plus one	
fmt24dr	fmt32dr	LOG	base 10 logarithm	
fmt24dr	fmt32dr	EXP2M1	base 2 exponentation minus one	
fmt24dr	fmt32dr	EXP2	base 2 exponentation	
fmt24dr	fmt32dr	EXPM1	base E exponentation minus one	
fmt24dr	fmt32dr	EXP	base E exponentation	
fmt24dr	fmt32dr	EXP10M1	base 10 exponentation minus one	
fmt24dr	fmt32dr	EXP10	base 10 exponentation	
fmt24dr	fmt32dr	SIN	trig functions in radians	
fmt24dr	fmt32dr	COS	trig functions in radians	
fmt24dr	fmt32dr	TAN	trig functions in radians	
fmt24dr	fmt32dr	ASIN	trig functions in radians	
fmt24dr	fmt32dr	ACOS	trig functions in radians	
fmt24dr	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	

Triple Operand Instructions, typically 32 or 40-bit instructions, see also RTF64 by Robert Finch

	fmt32drst, fmt32drnt	MEDIAN
	fmt32drst, fmt32drnt	MAX3
	fmt32drst, fmt32drnt	MIN3
	fmt32drst, fmt32drnt	ADD3

fmt32drst, fmt32drnt	INTP-BLEND-LERP	linear interpolation
fmt32drst, fmt32drnt	FMAC	multiply-add
fmt32drst, fmt32drnt	LUT	Boolean predicate over multiple bits (5 to 8), R & T are start and ending register numbers which s
fmt32drst, fmt32drnt	CMOV	Select one of two operands
fmt32drst, fmt32drnt	MERG	Select bits from two operands