A. Uht 4/3/1999

ICED Canned CPU Instruction Set Description

Group	Name	Description	Mnem.	Arguments	RTL	instruction format
						5 0 opcode major | 5 0 opcode minor
						31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0
						r1
1-a	logical	bitwise and	and	r1, r2, r3	r1<- r2 AND r3	1 0 0 0 0 1 | 0 1 | 0 0 | reserved | r3 | r2 | r1
		bitwise or	or	r1, r2, r3	r1<- r2 OR r3	0 1
		bitwise exclusive or	xor	r1, r2, r3	r1<- r2 XOR r3	1 0
		bitwise 1's complement	not	r1, r2	r1<- ~r2	1 1 | reserved
						ar log sl sr
1-b	shift	shift left	shl	r1, r2	r1<- ((r2 << 1), 0)	1 0 0 0 0 1 | 1 0 | 0 1 1 0 | reserved | r2 | r1
		shift right logical	shrl	r1, r2	r1<- (0, (r2 >> 1))	0 1 0 1
		shift right arithmetic	shra	r1, r2	r1<- (r2[31], (r2 >> 1))	1 0 0 1
						sb ad vt ct ~t
2	arithm.	add	add	r1, r2, r3	r1<- r2 + r3	1 0 0 0 1 0 | 0 1 0 0 1 | reserved | r3 | r2 | r1
		add and trap if carry	addct	r1, r2, r3	"; if Carry, TRAP	0 1 0 1 0
		add and trap if 2's complement overflow	addvt	r1, r2, r3	"; if oVerflow, TRAP	0 1 1 0 0
		subtract	sub	r1, r2, r3	r1<- r2 - r3	1 0 0 0 1
		subtract and trap if carry	subct	r1, r2, r3	"; if Carry, TRAP	1 0 0 1 0
		sub. and trap if 2's complement overflow	subvt	r1, r2, r3	"; if oVerflow, TRAP	1 0 1 0 0
						f
3	cnd. br.	conditional branch; non-zero or true test	brt	r2, rel	if (r2 != 0){PC <- PC + 4 + rel}	0 0 0 1 0 0 | 0 | rd | relative word address | r2 | reserved
		conditional branch; zero or false test	brf	r2, rel	if (r2 = 0){PC <- PC + 4 + rel}	1
4-a	br. -lnk	branch-and-link	brl	r1, addr	PC <- addr; r1 <- PC + 4	1 0 1 0 0 0 | absolute word address | r1
4-b	ind. br.	indirect branch	bri	r2	PC <- r2	0 0 1 0 0 0 | 0 | reserved | r2 | reserved
		return from interrupt	rti	r2	PC <- r2; re-enable interrupts	1
						u
5-a	ld. imm.	load immediate sign ext. LS 16 bits	ldi	r1, const	r1<- (others=>const[15], const)	1 1 0 0 0 0 | 0 | 0 | constant | reserved | r1
		load immediate upper (MS) 16 bits	ldiu	r1, const	r1<- (const, r1[15 downto 0]) Note: r1 also used as r2 source	1 1 0 0 0 0 | 0 | 1 | constant | r1 | r1
						w b
5-b	load	load word	ld	r1, off(r2)	r1<- M[off + r2]	1 1 0 0 0 0 | 1 | 1 0 | offset | reserved | r2 | r1
		load byte	ldb	r1, off(r2)	r1<-(others=>'0', M[off+r2][7 dt. 0])	0 1
						w b
5-c	store	store word	st	off(r2), r3	M[off + r2] <- r3	0 1 0 0 0 0 | 0 | 1 0 | offset | r3 | r2 | reserved
		store byte	stb	off(r2), r3	M[off+r2][7 dt. 0]<- r3[7 downto 0]	0 1
6	control	halt	halt	-	drain pipe, disable CPU	0 0 0 0 0 0 | 1 1 | reserved

NOTES and abbreviations

1.	TRAP	IR <- (brl R14, 0x4); --force execution of a brl instruction
2.	rd	reserved
3.	offset	byte offset; if word access, two LSB's must be 0 (aligned accesses)
4.	M[]	main memory access; 24 bit address space = 16 Mbytes
5.		for true unconditional branch, load r2 with the address and execute a "bri r2"
6.	ar	arithmetic
7.	log	logical
8.	sl	shift left
9.	sr	shift right
10.	sb	subtract
11.	ad	add
12.	vt	2's complement overflow trap
13.	ct	carry (1's complement overflow) trap
14.	~t	no trap (no overflow testing)
15.	f	false
16.	u	upper
17.	w	word
18.	b	byte
19.	r1	when bit 31 is a '1', then r1 is stored in the execution of the instruction