ULNAv1 Instruction Set, by Alphabetical Order

Instruction	Instruction Type	Instruction Number	RTN (PC = PC + 1, unless otherwise specified)	Updates Cond Codes
add	R	1	R[W] = R[A] + R[B]	X
addi	D	9	R[W] = R[A] + SignExtend(Imm5)	X
and	R	2	R[W] = R[A] AND R[B]	
andi	I	18	R[W] = R[W] AND SignExtend(Imm8)	
b	B	24	PC = PC + SignExtend(Imm11)	
b.eq	B	26	PC = PC + SignExtend(Imm11) if Z == 1	
b.ge	B	27	PC = PC + SignExtend(Imm11) if N == V	
b.gt	B	25	PC = PC + SignExtend(Imm11) if Z == 0 AND N == V	
b.le	B	30	PC = PC + SignExtend(Imm11) if NOT (Z == 0 AND N == V)	
b.lt	B	28	PC = PC + SignExtend(Imm11) if NOT (N == V)	
b.ne	B	29	PC = PC + SignExtend(Imm11) if Z == 0	
br	R	4	PC = R[A]	
call	I	23	R[W] = PC + 1; PC = PC + SignExtend(Imm8)	
cmp	I	21	Update Cond Codes by performing R[W] - SignExtend(Imm8)	X
halt	B	31	MemWord[0xFFFF] = 0xFFFF; PC = PC	
inci	I	17	R[W] = R[W] + SignExtend(Imm8)	X
ldw	R	7	R[W] = MemWord[R[A] + R[B]]	
ldwi	D	15	R[W] = MemWord[R[A] + SignExtend(Imm5)]	
movi	I	20	R[W] = ZeroExtend(Imm8)	
movis	I	22	R[W] = R[W] OR LogicalShiftLeft(ZeroExtend(Imm8), 8)	
neg	R	6	R[W] = NOT R[A]	
or	R	0	R[W] = R[A] OR R[B]	
ori	I	16	R[W] = R[W] OR ZeroExtend(Imm8)	
rot	D	14	R[W] = RollRight(R[A], Signed(Imm5))	
sha	D	8	R[W] = ArithmeticShiftRight(R[A], Signed(Imm5))	
shl	D	10	R[W] = LogicalShiftRight(R[A], Signed(Imm5))	
stw	R	3	MemWord[R[A] + R[B]] = R[W]	
stwi	D	11	MemWord[R[A] + SignExtend(Imm5)] = R[W]	
sub	R	5	R[W] = R[A] - R[B]	X
undef - 12	D	12	don’t care	
undef - 13	D	13	don’t care	
undef - 19	I	19	don’t care	

ULNAv1 Instruction Set, by Alphabetical Order

Instruction Number	**Instruction**
0	or
1	add
2	and
3	stw
4	br
5	sub
6	neg
7	ldw
8	sha
9	addi
10	shl
11	stwi
12	undef - 12
13	undef - 13
14	rot
15	ldwi
16	ori
17	inci
18	andi
19	undef - 19
20	movi
21	cmp
22	movis
23	call
24	b
25	b.gt
26	b.eq
27	b.ge
28	b.lt
29	b.ne
30	b.le
31	halt