_mov(s1, s2) load s2 into s1

_movi(s1, 10) load 10 into s1

_add(s1, s2) load s1 + s2 into s1

_addc(s1, s2) load s1 + s2 + Carry into s1

_addi(s1, 10) load s1 + 10 into s1

_addic(s1, 10) load s1 + 10 + carry into s1

_sub(s1, s2) load s1 - s2 into s1

_subi(s1, 10) load s1 - 10 into s1

_subc(s1, s2) load s1 - s2 - carry into s1

_subic(s1, 10) load s1 - 10 - carry into s1

_and(s1, s2) load s1 & s2 into s1

_andi(s1, 8) load s1 & 00001000 into s1

_or(s1, s2) load s1 OR s2 into s1

_or(s1, 8) load s1 OR 00001000 into s1

_shr(s1, s2) s1 = s2: carry<<1 car=s2(7)

_shl(s1, s2) s1 = carry: s2>>1 car=s2(0)

_jump(#start)
- _jump(z, #start) branch if Zero = 1
- _jump(nz, #start) branch if Zero = 0
- _jump(c, #start) branch if Carry = 1
- _jump(nc, #start) branch if Carry = 0

_call(#start)
- _call(z, #start) call if Zero = 1
- _call(nz, #start) call if Zero = 0
- _call(c, #start) call if Carry = 1
- _call(nc, #start) call if Carry = 0

_RET Return from call

_cmp(s1, s2) compare s1 and s2 (s1 - s2)

_cmpi(s1, 10) compare s1 and 10 (s1 - 10)

_clrc Carry = 0

_clrz Zero = 0

_setc Carry = 1

_setz Zero = 1

_imp(s0, portaddress) s0 = protaddress

_exp(s0, portaddress) Portaddress = S0

_expi(c, portaddress) Portaddress = c

vchip A vision chip instruction

&vchip A vision chip instruction with keep