SIMPLE RISC
INSTRUCTION SET ARCHITECTURE

v.2.3

16 oct. 2013

1. REGISTER SET:

8 registers **16** bits wide: R0 ... R7

---

2. MEMORY:

RAM **16** kbits = 1024 words x **16** bit

3.1 INTEGER DATA FORMAT:

16 bit, signed, 2's complement

3.2 FLOATING POINT DATA FORMAT:

S: sign bit
mantissa: 10 bit absolute value but only 9 bits provided. MSB=1 is implicit.
exponent: 6 bit signed, 2's complement

$N = (-1)^S \cdot \text{mantissa} \cdot 2^{\text{exponent}}$

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
S	**exponent**						**mantissa**								

---

4. INSTRUCTION SET (not complete):

NOP

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

ADD op0 op1 op2

R[op0] = R[op1] + R[op2] (integer addition)

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
							op 0			**op 1**			**op 2**		

ADDF op0 op1 op2

R[op0] = R[op1] + R[op2] (floating point addition)

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
							op 0			**op 1**			**op 2**		

SUB op0 op1 op2

R[op0] = R[op1] - R[op2] (integer subtraction)

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
							op 0			**op 1**			**op 2**		

SUBF op0 op1 op2

R[op0] = R[op1] - R[op2] (floating point subtraction)

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
							op 0			**op 1**			**op 2**		

AND op0 op1 op2

R[op0] = R[op1] & R[op2]

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
							op 0			**op 1**			**op 2**		

OR op0 op1 op2

R[op0] = R[op1] | R[op2]

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
							op 0			**op 1**			**op 2**		

XOR op0 op1 op2

R[op0] = R[op1] ^ R[op2]

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
							op 0			**op 1**			**op 2**		

NAND op0 op1 op2

R[op0] = ~(R[op1] & R[op2])

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
							op 0			**op 1**			**op 2**		

NOR op0 op1 op2

R[op0] = ~(R[op1] | R[op2])

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
							op 0			**op 1**			**op 2**		

NXOR op0 op1 op2

R[op0] = ~(R[op1] ^ R[op2])

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
							op 0			**op 1**			**op 2**		

SHIFTR op0 op1 op2

R[op0] = R[op1] >> (op2 + 1)

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
							op 0			**op 1**			**op 2**		

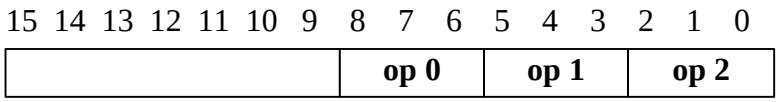
SHIFTRA op0 op1 op2

R[op0] = R[op1] >> (op2+1) (shift with sign extension)

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
							op 0			**op 1**			**op 2**		

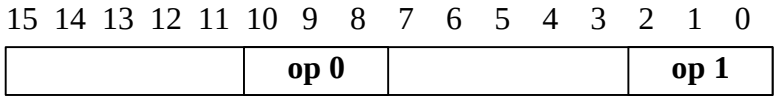
SHIFTL op0 op1 op2

R[op0] = R[op1] << (op2+1),



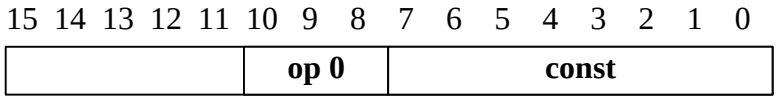
LOAD op0 op1

R[op0] = M[R[op1]]



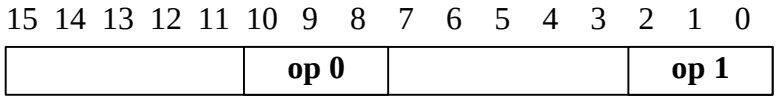
LOADC op0 #const

R[op0] = {R[op0][15:8], #const}



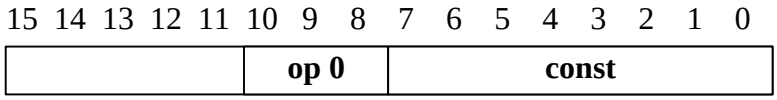
STORE op0 op1

M[R[op0]] = R[op1]



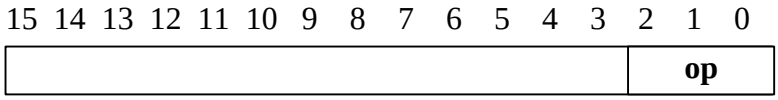
STOREC op0 #const

M[R[op0]] = {8'b0, #const}



JMP op

PC = R[op]



JMPR #offset

PC = PC + offset
offset: 6 bit signed, 2's complement

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
										offset					

JMPcond op0 op1

cond = **N/NN/Z/NZ**

if (cond(op0)) PC = R[op1] // for cond(op0) interpretation see $5

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
				cond			**op0**						**op1**		

JMPRcond op0 #offset

cond = **N/NN/Z/NZ**

if (cond(op0)) PC = PC + offset // for cond(op0) interpretation see $5
offset: 6 bit signed, 2's complement

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
				cond			**op0**			**offset**					

HALT

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

---

5. CONDITIONAL JUMP CONDITIONS:

condition	meaning
N(op0)	R[op0] < 0
NN(op0)	R[op0] >= 0
Z(op0)	R[op0] == 0
NZ(op0)	R[op0] != 0