SIMPLE RISC
INSTRUCTION SET ARCHITECTURE

v.3.2

1 oct. 2020

1. MAIN PARAMETERS

Data bus size: **D_BITS** (default value: 32)
Address bus size: **A_BITS** (default value: 10)

2. REGISTER SET:

8 general purpose registers, D_BITS wide: **R0 ... R7**

3. MEMORY:

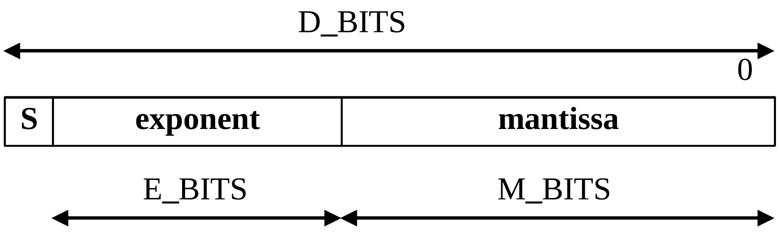
SRAM **MEMSIZE** words × D_BITS bits
where MEMSIZE = $2^{\text{A\_BITS}}$

4.1 INTEGER DATA FORMAT:

Signed, 2's complement

4.2 FLOATING POINT DATA FORMAT:

Complies with IEEE 754 Floating Point Standard
(default values E_BITS = 8, M_BITS = 23)



5. INSTRUCTION SET (not complete):

NOP

<table>
<tr><td>15</td><td>14</td><td>13</td><td>12</td><td>11</td><td>10</td><td>9</td><td>8</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr>
<tr><td colspan="16"></td></tr>
</table>

ADD op0 op1 op2

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

<table>
<tr><td>15</td><td>14</td><td>13</td><td>12</td><td>11</td><td>10</td><td>9</td><td>8</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr>
<tr><td colspan="7"></td><td colspan="3"><b>op 0</b></td><td colspan="3"><b>op 1</b></td><td colspan="3"><b>op 2</b></td></tr>
</table>

ADDF op0 op1 op2

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

<table>
<tr><td>15</td><td>14</td><td>13</td><td>12</td><td>11</td><td>10</td><td>9</td><td>8</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr>
<tr><td colspan="7"></td><td colspan="3"><b>op 0</b></td><td colspan="3"><b>op 1</b></td><td colspan="3"><b>op 2</b></td></tr>
</table>

SUB op0 op1 op2

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

<table>
<tr><td>15</td><td>14</td><td>13</td><td>12</td><td>11</td><td>10</td><td>9</td><td>8</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr>
<tr><td colspan="7"></td><td colspan="3"><b>op 0</b></td><td colspan="3"><b>op 1</b></td><td colspan="3"><b>op 2</b></td></tr>
</table>

SUBF op0 op1 op2

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

<table>
<tr><td>15</td><td>14</td><td>13</td><td>12</td><td>11</td><td>10</td><td>9</td><td>8</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr>
<tr><td colspan="7"></td><td colspan="3"><b>op 0</b></td><td colspan="3"><b>op 1</b></td><td colspan="3"><b>op 2</b></td></tr>
</table>

AND op0 op1 op2

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

<table>
<tr><td>15</td><td>14</td><td>13</td><td>12</td><td>11</td><td>10</td><td>9</td><td>8</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr>
<tr><td colspan="7"></td><td colspan="3"><b>op 0</b></td><td colspan="3"><b>op 1</b></td><td colspan="3"><b>op 2</b></td></tr>
</table>

OR op0 op1 op2

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

<table>
<tr><th>15</th><th>14</th><th>13</th><th>12</th><th>11</th><th>10</th><th>9</th><th>8</th><th>7</th><th>6</th><th>5</th><th>4</th><th>3</th><th>2</th><th>1</th><th>0</th></tr>
<tr><td colspan="7"></td><td colspan="3"><b>op 0</b></td><td colspan="3"><b>op 1</b></td><td colspan="3"><b>op 2</b></td></tr>
</table>

XOR op0 op1 op2

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

<table>
<tr><th>15</th><th>14</th><th>13</th><th>12</th><th>11</th><th>10</th><th>9</th><th>8</th><th>7</th><th>6</th><th>5</th><th>4</th><th>3</th><th>2</th><th>1</th><th>0</th></tr>
<tr><td colspan="7"></td><td colspan="3"><b>op 0</b></td><td colspan="3"><b>op 1</b></td><td colspan="3"><b>op 2</b></td></tr>
</table>

NAND op0 op1 op2

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

<table>
<tr><th>15</th><th>14</th><th>13</th><th>12</th><th>11</th><th>10</th><th>9</th><th>8</th><th>7</th><th>6</th><th>5</th><th>4</th><th>3</th><th>2</th><th>1</th><th>0</th></tr>
<tr><td colspan="7"></td><td colspan="3"><b>op 0</b></td><td colspan="3"><b>op 1</b></td><td colspan="3"><b>op 2</b></td></tr>
</table>

NOR op0 op1 op2

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

<table>
<tr><th>15</th><th>14</th><th>13</th><th>12</th><th>11</th><th>10</th><th>9</th><th>8</th><th>7</th><th>6</th><th>5</th><th>4</th><th>3</th><th>2</th><th>1</th><th>0</th></tr>
<tr><td colspan="7"></td><td colspan="3"><b>op 0</b></td><td colspan="3"><b>op 1</b></td><td colspan="3"><b>op 2</b></td></tr>
</table>

NXOR op0 op1 op2

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

<table>
<tr><th>15</th><th>14</th><th>13</th><th>12</th><th>11</th><th>10</th><th>9</th><th>8</th><th>7</th><th>6</th><th>5</th><th>4</th><th>3</th><th>2</th><th>1</th><th>0</th></tr>
<tr><td colspan="7"></td><td colspan="3"><b>op 0</b></td><td colspan="3"><b>op 1</b></td><td colspan="3"><b>op 2</b></td></tr>
</table>

SHIFTR op0 #val

R[op0] = R[op0] >> val

<table>
<tr><th>15</th><th>14</th><th>13</th><th>12</th><th>11</th><th>10</th><th>9</th><th>8</th><th>7</th><th>6</th><th>5</th><th>4</th><th>3</th><th>2</th><th>1</th><th>0</th></tr>
<tr><td colspan="7"></td><td colspan="3"><b>op 0</b></td><td colspan="6"><b>val</b></td></tr>
</table>

SHIFTRA op0 #val

R[op0] = R[op0] >> val (shift with sign extension)

<table>
<tr><td>15</td><td>14</td><td>13</td><td>12</td><td>11</td><td>10</td><td>9</td><td>8</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr>
<tr><td colspan="7"></td><td colspan="3"><b>op 0</b></td><td colspan="6"><b>val</b></td></tr>
</table>

SHIFTL op0 #val

R[op0] = R[op0] << val

<table>
<tr><td>15</td><td>14</td><td>13</td><td>12</td><td>11</td><td>10</td><td>9</td><td>8</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr>
<tr><td colspan="7"></td><td colspan="3"><b>op 0</b></td><td colspan="6"><b>val</b></td></tr>
</table>

LOAD op0 op1

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

<table>
<tr><td>15</td><td>14</td><td>13</td><td>12</td><td>11</td><td>10</td><td>9</td><td>8</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr>
<tr><td colspan="5"></td><td colspan="3"><b>op 0</b></td><td colspan="5">▨</td><td colspan="3"><b>op 1</b></td></tr>
</table>

Only the last A_BITS of R[op1] are used as memory address.

LOADC op0 #const

R[op0] = {R[op0][D_BITS-1:8], #const}

<table>
<tr><td>15</td><td>14</td><td>13</td><td>12</td><td>11</td><td>10</td><td>9</td><td>8</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr>
<tr><td colspan="5"></td><td colspan="3"><b>op 0</b></td><td colspan="8"><b>const</b></td></tr>
</table>

STORE op0 op1

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

<table>
<tr><td>15</td><td>14</td><td>13</td><td>12</td><td>11</td><td>10</td><td>9</td><td>8</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr>
<tr><td colspan="5"></td><td colspan="3"><b>op 0</b></td><td colspan="5">▨</td><td colspan="3"><b>op 1</b></td></tr>
</table>

Only the last A_BITS of R[op0] are used as memory address.

JMP op0

PC = R[op]

<table>
<tr><td>15</td><td>14</td><td>13</td><td>12</td><td>11</td><td>10</td><td>9</td><td>8</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr>
<tr><td colspan="4"></td><td colspan="9">▨</td><td colspan="3"><b>op0</b></td></tr>
</table>

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]

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

for cond(op0) interpretation see $5.1

JMPRcond op0 #offset

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

if (cond(op0)) PC = PC + offset

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

for cond(op0) interpretation see $5.1
offset: 6 bit signed, 2's complement

HALT

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

5.1. CONDITIONAL JUMP CONDITIONS:

cond code	cond	meaning of cond(op0)
000	N	R[op0] < 0
001	NN	R[op0] >= 0
010	Z	R[op0] == 0
011	NZ	R[op0] != 0
1xx	reserved	