# LD commands

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Instruction | Sub instruction | Register to write to (See Table 3) | Register to read from  (see table 4) | Not Used | Const/ram address as byte |
| 000 | 000 | 0000 | 000 | --- | 00000000 |

Table

LD: instruction mapping

|  |  |
| --- | --- |
| **Assembly** | **Binary** |
| LD A B | 000 000 |
| LD A &CONST | 000 001 |
| LD A &B | 000 001 |
| LD &CONST A | 000 010 |
| LD &A B | 000 010 |
| LD A CONST | 000 011 |
| CLR | 000 100 |
|  |  |

Table

Register to write to

NB this controls EW lines and MUX 3.1

|  |  |
| --- | --- |
| **Assembly** | **Binary** |
| Write A | 1000 |
| Write B | 0100 |
| Write C | 0010 |
| Write D | 0001 |
| Write A and C | 1010 |

Table

Register to read from

NB binary is encoding to set mux 1 and 2

|  |  |
| --- | --- |
| **Assembly** | **Binary** |
| Read From A | 010 |
| Read From B | 001 |
| Read From C | 011 |
| Read From D | 100 |

Table

# Addition Operations

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Instruction | Sub instruction | Register to write to (See Table 3) | Register 1 to read from  (see table 4) | Register 2 to read from  (see table 4) | Const as byte |
| 001 | 000 | 0000 | 000 | 000 | 00000000 |

Table 5

Addition: instruction mapping

|  |  |
| --- | --- |
| **Assembly** | **Binary** |
| ADD A B C | 001 000 |
| ADD A B Const | 001 001 |
| SUB A B C | 001 010 |
| SUB A B Const | 001 011 |
| LSH A B | 001 100 |
| RSH A B | 001 101 |
| LSH2S A B | 001 110 |
| RSH2S A B | 001 111 |

Table 6

# Logical operations

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Instruction | Sub instruction | Register to write to (See Table 3) | Register 1 to read from  (see table 4) | Register 2 to read from  (see table 4) | Const as byte |
| 010 | 000 | 0000 | 000 | 000 | 00000000 |

Table 7

Logical operations: instruction mapping

|  |  |
| --- | --- |
| **Assembly** | **Binary** |
| AND A B C | 010 000 |
| AND A B Const | 010 001 |
| OR A B C | 010 010 |
| OR A B Const | 010 011 |

Table 8

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Name | Instruction | Sub instruction | Register to write to (See Table 3) | Not used | Register 1 to read from  (see table 4) | Not Used |
| NOT A B | 010 | 100 | 0000 | --- | 000 | -------- |
| NOT2S A B | 010 | 111 | 0000 | --- | 000 |  |

Table 9

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Name | Instruction | Sub instruction | Register to write to (See Table 3) | Register 1 to read from  (see table 4) | Not Used | Not Used |
| ROL A B | 010 | 101 | 0000 | 000 | --- | -------- |
| ROR A B | 010 | 110 | 0000 | 000 | --- | -------- |

Table 10

# Logical Comparison Operations

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Instruction | Sub instruction | Register to write to (See Table 3) | Register 1 to read from  (see table 4) | Register 2 to read from  (see table 4) | Const as byte |
| 011 | 000 | 0000 | 000 | 000 | 00000000 |

Table 11

Logical comparisons: instruction mapping

NB these store result as 1 or zero in specified register

|  |  |
| --- | --- |
| **Assembly** | **Binary** |
| GR A B C | 011 000 |
| GR A B Const | 011 001 |
| GEQ A B C | 011 010 |
| GEQ A B Const | 011 011 |
| EQ A B C | 011 100 |
| EQ A B Const | 011 101 |

Table 12

# Misc Operations

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Name | Instruction | Sub instruction | Not used | Not used | Not used | Const as byte |
| PCU | 100 | 000 | ---- | --- | --- | 00000000 |
| PCL | 100 | 001 | ---- | --- | --- | 00000000 |
| LDI | 100 | 010 | ---- | --- | ---- | -------- |

Table 13

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Name | Instruction | Sub instruction | Not used | Register 1 to read from  (see table 4) | Not used | Const as byte |
| JNZ | 100 | 011 | ---- | 000 | --- | 00000000 |

Table 14

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Name | Instruction | Sub instruction | Not used | Not used | Not used | Not used |
| NOP | 100 | 100 | ---- | --- | --- | -------- |

Table 15

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Name | Instruction | Sub instruction | Not used | Const as 6 bits | Const as byte |
| OCL | 100 | 101 | ---- | 000000 | 00000000 |

Table 16

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Name | Instruction | Sub instruction | Not used | Set 1 for disable | Not used | Not used |
| SIR Const | 100 | 111 | ---- | --0 | --- | -------- |

Table 17

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Name | Instruction | Sub instruction | Not used | Not used | Not used | Const |
| WPR Const | 101 | 000 | ---- | --- | --- | 00000000 |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Name | Instruction | Sub instruction | Register to write to (See Table 3) | Not used | Not used | Not used |
| RPR A | 101 | 001 | 0000 | --- | --- | -------- |

Issues

1. Implement new codes

SIR CONST – enables or disable interrupts depending if const is 1 or 0

WPR Const – write new page register value

RPR Const – read current page register value

ROL A B – roll b to the left (10101 becomes 01011)

ROR A B – roll b to the right

LSH2S – shift left as 2s complement

RSH2S – shift right as 2s complement

NOT2S – invert the number (not the bits) equivalent to multiplying by -1

1. Implement new functionality in logisim

Read OCL register

Clear registers using clear pins

1. Compiler ignores blank lines
2. Change muxs to 0th index is ground