| 1st Byte | Instruction | Example | 128 | 8 | 64 | 32 | 16 | 8 | 4 2 | 1 | 2nd Byte | 3rd Byte | 4th Byte | | | |
|--|--|--|-------|---|-------------|--|--|--|---|--|------------------------|----------------------------|-------------------|-----|---|---|
| None Immediate | | | | 0 | 0 | | | | | | Address | Address | Target Address | | | |
| 2nd Immediate | | addji 1 A B | | | | | | | | | Literal | Address | Target Address | | | |
| | | | 1 | 1 | 0 | | | | | | | | larget Address | | | |
| 3rd Immediate | J | sublj A 1 A | | 0 | 1 | | | | | | Address | Literal | Target Address | | | |
| | | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | | |
| Operation | | | | | | 0 | 0 | | | | | | | | | |
| | | | | | | U | U | | | | | | | | | |
| Maths | | | | | | | | 0 | | | | | | | | |
| Add | add | add ABC | | | | | | 0 | 0 0 | 0 | Address | Address | Target Address | | | |
| Subtract | sub | sub A B C | | | | | | 0 | 0 0 | 1 | Address | Address | Target Address | | | |
| Multiply | mul | mul A B C | | | | | | 0 | 0 1 | 0 | Address | Address | Target Address | | | |
| Shift Left | | shi A B C | | | | | | | | | 71001000 | radicas | runget / ruur eus | | | |
| | shl | | | | | | | 0 | 0 1 | 1 | | | | | | |
| Shift Right | shr | shr A B C | | | | | | 0 | 1 0 | 0 | | | | | | |
| Negate | neg | neg A_C | | | | 0 | | 0 | 1 0 | 1 | | | | | | |
| Divide | div | div A B C | | | | | | 0 | 1 1 | 0 | Address | Address | Target Address | | | |
| Modulo | mod | mod A B C | | | | | | 0 | | | | | 10.9011100-000 | | | |
| Modulo | mod | MOUABC | | | | | | U | | 1 | | | | | | |
| | | | | | | | | | | | | | | | | |
| Bitwise | | | | | | 0 | | 1 | | | | | | | | |
| AND | and | and ABC | | | | 0 | | 1 | 0 0 | 0 | Address | Address | Target Address | | | |
| OR | or | or A B C | | | | | | 1 | 0 0 | 1 | Address | Address | Target Address | | | |
| | | | | | | | | | | | | Addiese | | | | |
| NOT | not | not A _ C | | | | | | 1 | 0 1 | 0 | Address | _ | Target Address | | | |
| XOR | xor | xor A B C | | | | 0 | | 1 | 0 1 | 1 | Address | Address | Target Address | | | |
| NAND | nand | nand A B C | | | | 0 | 0 | 1 | 1 0 | 0 | | | | | | |
| NOR | nor | nor A B C | | | | 0 | 0 | 1 | 1 0 | 1 | | | | | | |
| XNOR | xnor | xnor A B C | | | | 0 | 0 | 1 | 1 | 0 | | | | | | |
| ANUR | ANUI | AIU ABC | | | | U | U | | . 1 | · | | | | | | |
| | | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | | |
| Condition | | | | | | 0 | 1 | | | | | | | | | |
| Compare against values | | | | | | 0 | 1 | 1 | | | | | | | | |
| | | | | | | | | | | | | | | | | |
| Equal | je | je A B label_name | | | | | | | 0 0 | 1 | Address | Address | Jump Address | | | |
| Not Equal | jne | jne A B label_name | | | | | 1 | | 1 0 | 1 | Address | Address | Jump Address | | | |
| Less Than | į. | jl A B label_name | | | | 0 | 1 | | 0 1 | 0 | Address | Address | Jump Address | | | |
| Less Than Or Equal To | ile | jle A B label_name | | | | 0 | | | 0 1 | 1 | Address | Address | Jump Address | | | |
| | p~ | Jack Diebel anne | | | | 0 | | | 4 | 1 | | | | | | |
| Greater Than | 19 | jg A B label_name | | | | | 1 | | 1 1 | 1 | Address | Address | Jump Address | | | |
| Greater Than Or Equal To | jge | jge A B label_name | | | | 0 | 1 | | 1 1 | 0 | Address | Address | Jump Address | | | |
| Space for more | | | | | | | 1 | | | | Address | Address | Jump Address | | | |
| Always | jmp | jmplabel_name | | | | 0 | 1 | 0 | 1 1 | 1 | | | Jump Address | | | |
| 7411435 | J. P | Jinp label_liane | | | | | | | | | - | - | oump riourcoo | | | |
| | | | | | | | | | | | | | | | | |
| Moving | | | | U | 0 | 1 | U | | 0 | U | Address | - | Target Address | | | |
| Сору | сру | cpy A_B | | | | | | 0 | 0 | | | | | | | |
| Move (equivalent to copy) | mov | mov A_B | | | | | | 0 | 1 | | | | | | | |
| Stack Pop | рор | popA | | | | | | 4 | 0 | | | | | | | |
| | pop | popx | | | | | | | | | | | | | | |
| Stack Push | psh | psh A | | | | | | 1 | 1 | | | | | | | |
| | | | | | | | | | | | | | | | | |
| Function | | | | | | 1 | 0 | | 1 | | | | | | | |
| call | call | call label_name | | | | | | | | | | | | | | |
| | | | | | | | | | 0 1 | 0 | | | | | | |
| cai | | | | | | | | 0 | 0 1 | 0 | | | | | | |
| return | ret | ret | | | | | | 0 | 0 1 | 0 | | | | | | |
| return | | ret | | | | | | 0 | 0 1 | 0 | | | | | | |
| return | | ret | | | 1 | 1 | 0 | 0 | 0 1 | 0 | | Only 1 of these is require | ed | | | |
| return | ret | ret | | 0 | 1 | 1 | 1 | 0 | 0 1 | 0 | RAM Address | | | eq. | readl(ramNo.) idx to A | readframNo.) A.B.— read fidx in address A to (address B) |
| return RAM Read | ret | ret read 24 _ A | | 0 | 1 0 | 1 | 0 0 1 | 0 | 0 1 | 0 | RAM Address | Read Target Address | Target Address | eg. | readj(ramNo.) idx to A | read(j(ramNo.) _ A.B read (dx in address A) to (address B) with(ramNo.) A.B with(value in address A) to (first in address B) |
| return RAM Read Write | ret | ret read 24 _A writ A_ 24 | (| 0 | 1 0 1 | 1 | 0 0 1 1 | 0 | 0 1 | 0 1 | RAM Address Address | Read Target Address | | eg. | readl(ramNo.) idx to A writl(ramNo.) A to idx | read((ramNo.) _ A B read (idx in address A) to (address B) writt(ramNo.) A B write (value in address A) to (idx in address B) |
| return RAM Read Write RAM 0 | ret | red 24 _ A writ A _ 24 read) 24 _ A | (| 0 | 1 0 1 | 1 1 1 1 | 0 0 1 1 1 | 0 | 0 1 1 | 0 1 1 0 0 0 0 | | Read Target Address | Target Address | eg. | read((ramNo.) idx to A writl((ramNo.) A to idx | read((ramNo.) _A B read (ldx in address A) to (address B) writt((ramNo.) A B write (value in address A) to (ldx in address B) |
| return RAM Read Write RAM 0 RAM 1 | ret | read 24 _ A writ A 24 read() 024 _ A writ] 1 _ 24 | 6 | 0 | 1 0 1 | 1 1 1 1 1 1 | 0 0 1 1 1 1 1 | 0 0 0 0 | 0 1 0 1 | 0 1 0 0 1 1 | | Read Target Address | Target Address | eg. | read[(ramNo.) idx to A writl(ramNo.) A to idx | read((ramNo.) _ A B read (dx in address A) to (address B) writt((ramNo.) A B write (value in address A) to (dx in address B) |
| return RAM Read Write RAM 0 | ret | read 24 _ A writ A 24 read() 024 _ A writ] 1 _ 24 | 6 | 0 | 1 0 1 | 1 1 1 1 1 1 1 | 0 0 1 1 1 1 1 | 0 0 0 0 0 0 | 0 1 0 1 0 0 0 0 0 0 0 1 | 0 1 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 | | Read Target Address | Target Address | eg. | read((ramNo.) idx to A writt((ramNo.) A to idx | read((ramNo.) _ A B read (idx in address A) to (address B) wrttl((ramNo.) A B write (value in address A) to (dx in address B) |
| return RAM Read Write RAM 0 RAM 1 RAM 2 | ret | red 24 _ A writ A _ 24 read) 24 _ A | (| 0 0 0 | 1 0 1 | 1 1 1 1 1 1 1 1 | 0 0 1 1 1 1 1 1 1 | 0 | 0 1 0 1 | 0 1 0 1 1 1 1 1 1 | | Read Target Address | Target Address | eg. | read((ramNo.) idx to A writt((ramNo.) A to idx | read((ramNo.) _ A B read (idx in address A) to (address B) writt((ramNo.) A B write (value in address A) to (dx in address B) |
| return RAM Read Write RAM 0 RAM 1 RAM 2 RAM 3 | ret | read 24 _ A writ A 24 read() 024 _ A writ] 1 _ 24 | C | 0 0 0 | 1 0 1 | 1 | 0 0 1 1 1 1 1 1 1 | 0 | 0 1 0 1 0 0 0 0 0 0 0 1 | 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 | | Read Target Address | Target Address | eg. | read((ramNo.) idx to A writt((ramNo.) A to idx | read((ramNo.) _ A B read (ldx in address A) to (address B) writt((ramNo.) A B write (value in address A) to (ldx in address B) |
| return Read Write RAM 0 RAM 1 RAM 2 RAM 3 RAM 3 | ret | read 24 _ A writ A 24 read() 024 _ A writ] 1 _ 24 | C | 0 0 0 | ↓ 0 1 | 1 | 0 0 1 1 1 1 1 1 1 1 1 | 0 | 0 1 0 0 0 0 0 0 0 0 1 0 1 1 0 | 0 1 0 1 1 1 1 | | Read Target Address | Target Address | eg. | read((ramNo.) idx to A writt((ramNo.) A to idx | read((ramNo.) _ A B read (idx in address A) to (address B) writt(ramNo.) A B write (value in address A) to (idx in address B) |
| return RAM Read Write RAM 0 RAM 1 RAM 2 RAM 3 | ret | read 24 _ A writ A 24 read() 024 _ A writ] 1 _ 24 | (| 0 0 0 | 1 0 1 | 1 1 1 1 1 1 1 1 1 1 1 1 1 1 | 0 0 1 1 1 1 1 1 1 1 1 1 1 1 | 0 0 0 0 0 0 0 | 0 1 0 0 1 0 0 0 0 0 0 0 1 0 1 1 0 1 0 0 | 0 1 0 1 1 1 1 0 1 | | Read Target Address | Target Address | eg. | read((ramNo.) idx to A wrt((ramNo.) A to idx | read((ramNo.) _ A B read (idx in address A) to (address B) writt((ramNo.) A B write (value in address A) to (dx in address B) |
| return Read Write RAM 0 RAM 1 RAM 2 RAM 3 RAM 3 | ret | read 24 _ A writ A 24 read() 024 _ A writ] 1 _ 24 | C | 0 0 0 | 1 0 1 | 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 | 0 0 1 1 1 1 1 1 1 1 1 1 1 1 1 | 0 0 0 0 0 0 0 | 0 1 0 0 1 0 0 0 0 0 0 0 1 0 1 0 1 0 0 1 1 0 | 0 1 1 1 1 1 1 1 1 | | Read Target Address | Target Address | eg. | read((ramNo.) idx to A writt((ramNo.) A to idx | read((ramNo.) _AB read (idx in address A) to (address B) writt((ramNo.) AB write (value in address A) to (idx in address B) |
| return Read Write RAM 0 RAM 1 RAM 2 RAM 3 RAM 3 | ret | read 24 _ A writ A 24 read() 024 _ A writ] 1 _ 24 | C | 0 0 0 | 1 0 1 | 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 | 0 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 | 0 0 0 0 0 0 0 0 | 0 1 0 0 0 0 0 0 0 0 1 1 1 0 1 1 1 1 | 0 1 1 1 1 1 1 0 1 1 1 | | Read Target Address | Target Address | eg. | read((ramNo.) idx to A writ((ramNo.) A to idx | read((ramNo.) _ A B — read (idx in address A) to (address B) writt((ramNo.) A B _ — write (value in address A) to (dx in address B) |
| return Read Write RAM 0 RAM 1 RAM 2 RAM 3 RAM 3 | ret | read 24 _ A writ A 24 read() 024 _ A writ] 1 _ 24 | (| 0 0 0 | 1 0 1 | 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 | 0 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 | 0 0 0 0 0 0 0 0 | 0 1 0 1 0 0 0 0 0 0 0 1 1 1 0 1 1 1 1 1 1 1 | 0 1 1 1 1 1 1 1 1 1 1 1 | | Read Target Address | Target Address | eg. | read((ramNo.) idx to A writ((ramNo.) A to idx | read((ramNo.) _ A B read (dx in address A) to (address B) writt((ramNo.) A B writle (value in address A) to (dx in address B) |
| return Read Write RAM 0 RAM 1 RAM 2 RAM 3 RAM 3 | ret | read 24 _ A writ A 24 read() 024 _ A writ] 1 _ 24 | (| 0 0 0 | 1 0 1 | 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 | 0 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 | 0 0 0 0 0 0 0 0 0 0 0 1 1 1 | 0 1 1 0 0 0 0 0 0 0 1 1 0 1 1 1 1 1 1 0 | 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 | | Read Target Address | Target Address | eg. | read((ramNo.) idx to A writ((ramNo.) A to idx | read((ramNo.) _ A B — read (dx in address A) to (address B) writi((ramNo.) A B _ — write (value in address A) to (dx in address B) |
| return Read Write RAM 0 RAM 1 RAM 2 RAM 3 RAM 3 | ret | read 24 _ A writ A 24 read() 024 _ A writ] 1 _ 24 | C | 0000 | 1 0 1 | 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 | 0 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 | 0 0 0 0 0 0 0 0 0 0 0 0 | 0 1 1 0 0 0 0 0 0 0 0 1 1 0 0 1 1 1 1 1 | 0 1 1 0 1 1 1 1 0 0 1 1 1 1 1 | | Read Target Address | Target Address | eg. | read((ramNo.) idx to A. writ((ramNo.) A to idx | read((ramNo.) _A B read (idx in address A) to (address B) writt((ramNo.) A B write (value in address A) to (idx in address B) |
| return Read Write RAM 0 RAM 1 RAM 2 RAM 3 RAM 3 | ret | read 24 _ A writ A 24 read() 024 _ A writ] 1 _ 24 | (| 0 0 0 0 | 1 0 1 | 1 | 0 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 | 0 0 0 0 0 0 0 0 0 0 0 | 0 1 1 0 0 0 0 0 0 0 0 1 1 0 0 1 1 1 1 1 | 0 1 1 0 1 1 1 1 1 1 1 1 1 1 1 1 | | Read Target Address | Target Address | eg. | read((ramNo.) idx to A writt((ramNo.) A to idx | read((ramNo.) _ A B — read (idx in address A) to (address B) writi(ramNo.) A B _ — write (value in address A) to (dx in address B) |
| return Read Write RAM 0 RAM 1 RAM 2 RAM 3 RAM 3 | ret | read 24 _ A writ A 24 read() 024 _ A writ] 1 _ 24 | C | 0 0 0 0 | i 0 1 1 | 1 | 0 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 | 0 0 0 0 0 0 0 0 0 0 0 | 0 1 1 0 0 0 0 0 0 0 1 1 0 0 0 0 0 0 0 0 | 0 1 1 1 1 1 1 0 1 1 1 1 1 1 1 1 1 1 1 1 | | Read Target Address | Target Address | eg. | read((ramNo.) idx to A. writ((ramNo.) A to idx | read((ramNo.) _A B — read (dx in address A) to (address B) writt(ramNo.) A B _ — write (value in address A) to (dx in address B) |
| return Read Write RAM 0 RAM 1 RAM 2 RAM 3 RAM 3 | ret | read 24 _ A writ A 24 read() 024 _ A writ] 1 _ 24 | C | 00000 | 1 0 1 | 1 | 0 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 | 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 | 0 1 1 0 0 0 0 0 0 0 0 1 1 0 0 0 0 0 0 0 | 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 | | Read Target Address | Target Address | eg. | read((ramNo.) idx to A writt(ramNo.) A to idx | read((ramNo.) _ A B — read ((dx in address A) to (address B) writl(ramNo.) A B _ — writle (value in address A) to (dx in address B) |
| return Read Write RAM 0 RAM 1 RAM 2 RAM 3 RAM 3 | ret | read 24 _ A writ A 24 read() 024 _ A writ] 1 _ 24 | C | 0000 | 1 0 0 1 | 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 | 0 0 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 | 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 | 0 1 1 0 1 1 1 1 1 0 0 0 1 1 0 0 1 1 0 0 0 0 1 1 0 0 0 1 1 0 0 0 1 1 0 0 0 0 1 1 0 0 0 0 1 1 0 0 0 0 1 1 0 0 0 0 0 1 1 0 0 0 0 0 1 1 0 0 0 0 0 1 1 0 0 0 0 0 1 1 0 0 0 0 0 0 1 1 0 | 0 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 | | Read Target Address | Target Address | eg. | read((ramNo.) idx to A wrt((ramNo.) A to idx | read((ramNo.) _ A B — read (dx in address A) to (address B) writ((ramNo.) A B _ — write (value in address A) to (dx in address B) |
| return Read Write RAM 0 RAM 1 RAM 2 RAM 3 RAM 3 | ret | read 24 _ A writ A 24 read() 024 _ A writ] 1 _ 24 | C C | 0 | 1 0 1 | 1 | 0 0 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 | 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 | 0 1 1 0 0 0 0 0 0 1 1 0 0 0 0 0 0 1 1 0 0 0 0 0 1 1 1 0 0 1 1 1 0 0 1 1 1 0 0 0 0 0 1 1 1 0 0 1 1 1 0 0 0 0 1 1 1 0 0 1 | 0 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 | | Read Target Address | Target Address | eg. | read((ramNo.) idx to A writ((ramNo.) A to idx | read((ramNo.) _ A B — read ((dx in address A) to (address B) writt((ramNo.) A B _ — write (value in address A) to (dx in address B) |
| return RAM Read Write RAM 0 RAM 1 RAM 1 RAM 2 RAM 3 RAM 4 RAM 5 RAM 6 RAM 6 | ret | read 24 _ A writ A 24 read() 024 _ A writ] 1 _ 24 | C C | 0 0 0 0 | 1 0 1 | 1 | 0 0 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 | 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 1 | 0 1 1 0 1 1 1 0 1 1 1 1 1 1 1 1 1 1 1 1 | 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 | | Read Target Address | Target Address | eg. | read((ramNo.) idx to A writ((ramNo.) A to idx | read((ramNo.) _ A B — read (dx in address A) to (address B) writt((ramNo.) A B _ — write (value in address A) to (dx in address B) |
| return Read Write RAM 0 RAM 1 RAM 2 RAM 3 RAM 3 | ret | read 24 _ A writ A 24 read() 024 _ A writ] 1 _ 24 | C C | 00000 | 1 0 1 | 1 | 0 0 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 | 0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 1 1 | 0 1 1 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 | 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 | | Read Target Address | Target Address | eg. | read((ramNo.) idx to A writt(ramNo.) A to idx | read((ramNo.) _ A B read (idx in address A) to (address B) writt(ramNo.) A B write (value in address A) to (idx in address B) |
| return RAM Read Write RAM 0 RAM 1 RAM 1 RAM 2 RAM 3 RAM 4 RAM 5 RAM 5 RAM 6 RAM 6 RAM 6 RAM 6 RAM 7 | ret | read 24 _ A writ A 24 read() 024 _ A writ] 1 _ 24 | C | 0 0 0 0 0 | 1 0 1 | 1 | 0 0 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 | 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 1 | 0 1 1 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 | 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 | | Read Target Address | Target Address | eg. | read((ramNo.) idx to A writ((ramNo.) A to idx | read((ramNo.) _A B — read (dx in address A) to (address B) writi(ramNo.) A B _ — write (value in address A) to (dx in address B) |
| return RAM Read Write RAM 0 RAM 1 RAM 1 RAM 2 RAM 3 RAM 4 RAM 5 RAM 6 RAM 6 | ret | read 24 _ A writ A 24 read() 024 _ A writ] 1 _ 24 | C C | 000000000000000000000000000000000000000 | 1 0 1 | 1 | 0 0 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 | 0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 1 1 | 0 1 1 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 | 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 | | Read Target Address | Target Address | eg. | read((ramNo.) idx to A writ((ramNo.) A to idx | read((ramNo.) _AB read (idx in address A) to (address B) writt(ramNo.) AB write (value in address A) to (idx in address B) |
| return RAM Read Write RAM 0 RAM 1 RAM 1 RAM 2 RAM 3 RAM 4 RAM 5 RAM 5 RAM 6 RAM 6 RAM 6 RAM 6 RAM 7 | ret | read 24 _ A writ A 24 read() 024 _ A writ] 1 _ 24 | C | 0 | 1 0 1 | 1 | 0 0 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 | 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 1 | 0 1 1 0 1 1 1 1 1 1 1 1 1 | 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 | | Read Target Address | Target Address | eg. | read((ramNo.) idx to A writt((ramNo.) A to idx | read((ramNo.) _ A B — read (idx in address A) to (address B) writl(ramNo.) A B _ — write (value in address A) to (dx in address B) |
| return RAM Read Write RAM 0 RAM 1 RAM 2 RAM 2 RAM 3 RAM 4 RAM 5 RAM 5 RAM 6 RAM 6 RAM 6 RAM 6 RAM 7 RAM 8 RAM 9 RAM 9 RAM 9 RAM 9 RAM 9 RAM 9 RAM 15 | red read writ | red read 24 _A writ A _ 24 read(0 24 _A writ(1 A _ 24 writ(2 lo A _ | C | 0 0 0 0 0 | 1 0 1 1 | 1 | 0 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 | 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 | 0 1 1 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 | 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 | | Read Target Address | Target Address | eg. | read((ramNo.) idx to A wrt((ramNo.) A to idx | read((ramNo.) _ A B — read (dx in address A) to (address B) writt((ramNo.) A B _ — write (value in address A) to (dx in address B) |
| return RAM Read Write RAM 0 RAM 1 RAM 1 RAM 2 RAM 3 RAM 4 RAM 5 RAM 5 RAM 6 RAM 6 RAM 6 RAM 6 RAM 7 | red read writ | red read 24 _A writ A _ 24 read(0 24 _A writ(1 A _ 24 writ(2 lo A _ | c | 0000 | 1 0 1 | | 0 0 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 | 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 | 0 1 1 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 | 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 | | Read Target Address | Target Address | eg. | read((ramNo.) idx to A writ((ramNo.) A to idx | read((ramNo.) _ A B — read (idx in address A) to (address B) writl(ramNo.) A B _ — write (value in address A) to (dx in address B) |
| return RAM Read Write RAM 0 RAM 1 RAM 2 RAM 2 RAM 3 RAM 4 RAM 5 RAM 5 RAM 6 RAM 6 RAM 6 RAM 6 RAM 6 RAM 7 RAM 8 RAM 6 RAM | read writ read writ final fi | red 24_A writ A_24 read(0 24_A writ(1 A_24 writ(2 to A_ writ(2 to A_ writ(3 to A_ w | c | 0 0 0 0 0 | 1 0 1 | 1 | 0 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 | 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 | 0 1 1 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 | 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 | | Read Target Address | Target Address | eg. | read((ramNo.) idx to A writ((ramNo.) A to idx | read((ramNo.) _ A B — read (dx in address A) to (address B) writt((ramNo.) A B _ — write (value in address A) to (dx in address B) |
| return RAM Read Write RAM 0 RAM 1 RAM 2 RAM 3 RAM 3 RAM 5 RAM 6 RAM 6 RAM 1 RAM 5 RAM 6 RAM 6 RAM 1 RAM 15 RAM 15 RAM 15 RAM 15 Running the Processor Unfortunately, I was unable to im | ret read writ nglement 16-bit data, so 255 is cessor both have double-dabbit | red 24_A writ A_24 read(0 24_A writ(1 A_24 writ(2 to A_ writ(2 to A_ writ(3 to A_ w | c | 0 | 1 0 1 | | 0 0 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 | 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 | 0 1 1 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 | 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 | | Read Target Address | Target Address | eg. | read((ramNo.) idx to A writ((ramNo.) A to idx | read((ramNo.) _ A B read (idx in address A) to (address B) writt(ramNo.) A B write (value in address A) to (dx in address B) |
| return RAM Read Write RAM 0 RAM 1 RAM 2 RAM 3 RAM 3 RAM 5 RAM 6 RAM 6 RAM 1 RAM 5 RAM 6 RAM 6 RAM 1 RAM 15 RAM 15 RAM 15 RAM 15 Running the Processor Unfortunately, I was unable to im | ret read writ nglement 16-bit data, so 255 is cessor both have double-dabbit | red 24_A writ A_24 read(0 24_A writ(1 A_24 writ(2 to A_ writ(2 to A_ writ(3 to A_ w | c | 0 | 1 0 1 | 1 | 0 0 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 | 0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 1 1 | 0 1 1 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 0 0 0 0 0 1 1 0 0 0 0 1 1 1 0 0 0 0 0 1 1 1 0 0 1 1 1 1 0 0 1 | 0 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 | | Read Target Address | Target Address | eg. | read((ramNo.) idx to A writ((ramNo.) A to idx | read((ramNo.) _ A B read (idx in address A) to (address B) writi(ramNo.) A B write (value in address A) to (dx in address B) |
| return RAM Read Write RAM 0 RAM 1 RAM 2 RAM 2 RAM 3 RAM 4 RAM 5 RAM 5 RAM 6 RAM 6 RAM 6 RAM 6 RAM 6 RAM 7 RAM 8 RAM 6 RAM | ret read writ nglement 16-bit data, so 255 is cessor both have double-dabbit | red 24_A writ A_24 read(0 24_A writ(1 A_24 writ(2 to A_ writ(2 to A_ writ(3 to A_ w | c | 0 | 1 0 1 | 1 | 0 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 | 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 | 0 1 1 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 | 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 | | Read Target Address | Target Address | eg. | read((ramNo.) idx to A wrt((ramNo.) A to idx | read((ramNo.) _ A B — read (idx in address A) to (address B) writt((ramNo.) A B _ — write (value in address A) to (dx in address B) |
| return RAM Read Write RAM 0 RAM 1 RAM 2 RAM 3 RAM 3 RAM 4 RAM 5 RAM 5 RAM 5 RAM 5 RAM 6 RAM 1 RAM 6 RAM 15 RAM 15 RAM 15 Running the Processor Unfortunately, I was unable to im The Input and Output of the proc | rest read writ nplement 16-bit data, so 255 is cessor both have double-dabble, | red read 24 _A writ A_ 24 read() 24 A writ A_ 24 writ C to A_ writ N_ 24 | c c | 0 0 0 0 | 1 0 1 | 1 | 0 0 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 | 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 1 | 0 1 1 0 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 | 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 | | Read Target Address | Target Address | eg. | read((ramNo.) idx to A writ((ramNo.) A to idx | read((ramNo.) _ A B read (idx in address A) to (address B) writl(ramNo.) A B write (value in address A) to (dx in address B) |
| return RAM Read Write RAM 0 RAM 1 RAM 2 RAM 2 RAM 3 RAM 4 RAM 5 RAM 5 RAM 5 RAM 5 RAM 6 Use of the color of the proc of access when using the processor! When you open the processor! | ret read writ nplement 16-bit data, so 255 is cossor both have double-dabble he processor. | red 24 A writ A, 24 read 24 A writ A, 24 writ B T A writ A writ A writ B T | c c | 0 0 0 | 1 0 1 | 1 | 0 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 | 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 | 0 1 1 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 | 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 | | Read Target Address | Target Address | eg. | read((ramNo.) idx to A writ((ramNo.) A to idx | read((ramNo.) _ A B — read (dx in address A) to (address B) writt((ramNo.) A B _ — write (value in address A) to (dx in address B) |
| return RAM Read Write RAM 0 RAM 1 RAM 2 RAM 3 RAM 3 RAM 4 RAM 5 RAM 5 RAM 5 RAM 5 RAM 6 RAM 1 RAM 6 RAM 15 RAM 15 RAM 15 Running the Processor Unfortunately, I was unable to im The Input and Output of the proc | ret read writ nplement 16-bit data, so 255 is cossor both have double-dabble he processor. | red 24 A writ A, 24 read 24 A writ A, 24 writ B T A writ A writ A writ B T | | 0 | 1 0 1 | | 0 0 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 | 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 | 0 1 1 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 | 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 | | Read Target Address | Target Address | eg. | read((ramNo.) idx to A writ((ramNo.) A to idx | read((ramNo.) _ A B — read (idx in address A) to (address B) writl(ramNo.) A B _ — write (value in address A) to (dx in address B) |
| return RAM Read Write RAM 0 RAM 1 RAM 2 RAM 2 RAM 3 RAM 4 RAM 5 RAM 5 RAM 5 RAM 5 RAM 6 Use of the color of the proc of access when using the processor! When you open the processor! | ret read writ nplement 16-bit data, so 255 is cossor both have double-dabble he processor. | red 24 A writ A, 24 read 24 A writ A, 24 writ B T A writ A writ A writ B T | c | 0000 | 1 0 1 | | 0 0 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 | 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 | 0 1 1 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 1 1 | 0 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 | | Read Target Address | Target Address | eg. | read((ramNo.) idx to A writ((ramNo.) A to idx | read((ramNo.) _ A B — read (dx in address A) to (address B) writt((ramNo.) A B _ — write (value in address A) to (dx in address B) |
| return RAM Read Write RAM 0 RAM 1 RAM 2 RAM 2 RAM 3 RAM 4 RAM 5 RAM 5 RAM 6 Under the processor Unfortunately, I was unable to im The input and Output of the proc for ease of access when using in When you open the processor fland use the poke tool to press th | read writ nplement 16-bit data, so 255 is cessor both have double-dabble he processor. le, the first thing you should do | read 24 _A writ4_24 read 24 _A writ5_24 read(0.24 _A writ5_10_24 w | c | 0 | 1 0 1 | | 0 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 | 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 | 0 1 1 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 | 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 | | Read Target Address | Target Address | eg. | read((ramNo.) idx to A writ((ramNo.) A to idx | read((ramNo.) _ A B — read (idx in address A) to (address B) writt(ramNo.) A B _ — write (value in address A) to (dx in address B) |
| return RAM Read Write RAM 0 RAM 1 RAM 2 RAM 3 RAM 3 RAM 5 RAM 5 RAM 6 RAM 5 RAM 6 RAM 1 RAM 5 RAM 6 RAM 1 RAM 6 RAM 1 RAM 6 RAM 15 Running the Processor Unfortunately, I was unable to im The Input and Output of the proc for ease of access when using th | rest read writ mplement 16-bit data, so 255 is cessor both have double-dabbli he processor. le, the first thing you should do he PRESET pin. This primes the component labeled "Program" a | read 24 _A writ A_ 24 read 24 _A writ A_ 24 read 24 _A writ I A_ 24 writ I A_ 24 writ I C Io A_ the max right now - sorry. the max right now - sorry. timplementation is find the l/o area s RAMA banks. and edit the contents right-clicking | c | 0000 | 1 0 1 | | 0 0 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 | 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 | 0 1 1 0 0 0 0 0 0 0 0 0 0 1 1 1 1 0 0 1 1 1 1 1 1 1 1 1 1 | 0 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 | | Read Target Address | Target Address | eg. | read((ramNo.) idx to A writ((ramNo.) A to idx | read((ramNo.) _ A B read (idx in address A) to (address B) writt((ramNo.) A B write (value in address A) to (dx in address B) |
| return RAM Read Write RAM 0 RAM 1 RAM 2 RAM 2 RAM 3 RAM 4 RAM 5 RAM 5 RAM 6 RAM 5 RAM 6 The second | ret read writ nplement 16-bit data, so 255 is cessor both have double-dabble he processor. le, the first thing you should do he 'RESET' pin. This primes the component labelled 'Program' at it Contents'. There, you can di | read 24 _A writ A_ 24 read 24 _A writ A_ 24 read 24 _A writ I A_ 24 writ I A_ 24 writ I C Io A_ the max right now - sorry. the max right now - sorry. timplementation is find the l/o area s RAMA banks. and edit the contents right-clicking | c c | 00000 | 1 0 1 | | 0 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 | 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 1 | 0 1 1 0 0 0 0 0 0 0 1 1 0 0 0 0 1 1 1 0 0 0 0 0 1 | 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 | | Read Target Address | Target Address | eg. | read((ramNo.) idx to A wnt((ramNo.) A to idx | read((ramNo.) _ A B — read (dx in address A) to (address B) witl((ramNo.) A B _ — write (value in address A) to (dx in address B) |
| return RAM Read Write RAM 0 RAM 1 RAM 2 RAM 3 RAM 3 RAM 5 RAM 5 RAM 6 RAM 5 RAM 6 RAM 1 RAM 5 RAM 6 RAM 1 RAM 6 RAM 1 RAM 6 RAM 15 Running the Processor Unfortunately, I was unable to im The Input and Output of the proc for ease of access when using th | ret read writ nplement 16-bit data, so 255 is cessor both have double-dabble he processor. le, the first thing you should do he 'RESET' pin. This primes the component labelled 'Program' at it Contents'. There, you can di | read 24 _A writ A_ 24 read 24 _A writ A_ 24 read 24 _A writ I A_ 24 writ I A_ 24 writ I C Io A_ the max right now - sorry. the max right now - sorry. timplementation is find the l/o area s RAMA banks. and edit the contents right-clicking | c c | 0000 | 1 0 1 | | 0 0 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 | 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 1 | 0 1 1 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 | 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 | | Read Target Address | Target Address | eg. | read((ramNo.) idx to A writ((ramNo.) A to idx | read((ramNo.) _ A B read (idx in address A) to (address B) writi(ramNo.) A B write (value in address A) to (dx in address B) |
| return RAM Read Write RAM 0 RAM 1 RAM 2 RAM 2 RAM 3 RAM 5 RAM 5 RAM 5 RAM 6 RAM 15 RAM 15 Running the Processor Unfortunately, I was unable to im The Input and Output of the pro- for ease of access when using th When you open the processor fli and use the poke tool to press th Next, you should find the ROM o the component and pressing "Ed- program, or you can open a prog | read writ read writ higherment 16-bit data, so 255 is cessor both have double-dabble he processor. le, the first thing you should do he "RESET pin. This primes the component labelled "Proping the double-dabble for the processor. It Contents." There, you can di gram file from your PC. | read 24 _ A writ A_ 24 read 24 _ A writ 1 A_ 24 writ 1 A_ 24 writ 1 A_ 24 writ 2 A _ A writ 2 A _ A writ 3 A_ 24 writ 4 A_ 25 writ 5 A_ A writ 6 A _ A writ 6 A _ A writ 7 A_ 26 writ 7 A_ 26 writ 8 A _ A writ 1 A_ 26 writ 1 A_ | | 0000 | 1 0 1 | | 0 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 | 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 | 0 1 1 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 1 1 | 0 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 | | Read Target Address | Target Address | eg. | read((ramNo.) idx to A writ((ramNo.) A to idx | read((ramNo.) _ A B — read (dx in address A) to (address B) writt((ramNo.) A B _ — write (value in address A) to (dx in address B) |
| return RAM Read Write RAM 0 RAM 1 RAM 2 RAM 2 RAM 3 RAM 5 RAM 5 RAM 5 RAM 6 RAM 15 RAM 15 Running the Processor Unfortunately, I was unable to im The Input and Output of the pro- for ease of access when using th When you open the processor fli and use the poke tool to press th Next, you should find the ROM o the component and pressing "Ed- program, or you can open a prog | read writ read writ higherment 16-bit data, so 255 is cessor both have double-dabble he processor. le, the first thing you should do he "RESET pin. This primes the component labelled "Proping the double-dabble for the processor. It Contents." There, you can di gram file from your PC. | read 24 _ A writ A_ 24 read 24 _ A writ 1 A_ 24 writ 1 A_ 24 writ 1 A_ 24 writ 2 A _ A writ 2 A _ A writ 3 A_ 24 writ 4 A_ 25 writ 5 A_ A writ 6 A _ A writ 6 A _ A writ 7 A_ 26 writ 7 A_ 26 writ 8 A _ A writ 1 A_ 26 writ 1 A_ | c | 0000 | 1 0 1 | | 0 0 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 | 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 | 0 1 1 0 0 0 0 0 0 0 0 0 1 1 0 0 0 0 1 1 1 1 1 1 0 0 1 | 0 | | Read Target Address | Target Address | eg. | read((ramNo.) idx to A writ((ramNo.) A to idx | read((ramNo.) _ A B — read (idx in address A) to (address B) writl(ramNo.) A B _ — write (value in address A) to (dx in address B) |
| return RAM Read Write RAM 0 RAM 1 RAM 2 RAM 3 RAM 3 RAM 5 RAM 5 RAM 6 RAM 1 RAM 5 RAM 6 RAM 1 RAM 5 RAM 6 RAM 1 RA | read writ | read 24 _A writ A_ 24 read 24 _A writ A_ 24 readil 24 A writ (A_ 24 writ (A_ 24 writ (C to A_ writ (A_ 24 writ (A_ 24 writ (C to A_ writ (A_ 24 | c c | 00000 | 1 0 1 | | 0 0 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 | 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 | 0 1 1 0 0 0 0 0 1 1 1 1 0 0 1 1 1 1 1 1 | 0 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 | | Read Target Address | Target Address | eg. | read((ramNo.) idx to A writ((ramNo.) A to idx | read((ramNo.) _ A B — read (dx in address A) to (address B) writ((ramNo.) A B _ — write (value in address A) to (dx in address B) |
| return RAM Read Write RAM 0 RAM 1 RAM 2 RAM 2 RAM 3 RAM 4 RAM 5 RAM 5 RAM 6 RAM 15 RAM 6 RAM 15 Running the Processor Unfortunately, I was unable to im The Input and Output of the processor floor ease of access when using th When you open the processor floor ease of access when using th When you open the processor floor ease of access when using th When you open the processor floor ease of access when using th When you open the processor floor ease of access when using th When you open the processor floor ease of access when using th When you open the processor floor ease of access when using th Then, you should find the RCM of the component and pressing "Edprogram, or you can open a prog Then, you should set your input a program, in which case you i | rest read writ pplement 16-bit data, so 255 is cossor both have double-dabble he processor. le, the first thing you should do he "RESET pin. This primes the component labelled "Program" to Contents. There, you can di | read 24 _ A writ A_ 24 read 24 _ A writ A_ 24 read 24 _ A writ I A_ 24 writ I A_ 24 writ I A_ 25 writ I A_ 26 writ I A_ 26 read 26 _ A writ I A_ 26 | c c | 00000 | 1 0 1 | 1 | 0 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 | 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 1 | 0 1 1 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 1 | 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 | | Read Target Address | Target Address | eg. | read((ramNo.) idx to A wn(((ramNo.) A to idx | read((ramNo.) _ A B — read (dx in address A) to (address B) writ((ramNo.) A B _ — write (value in address A) to (dx in address B) |
| return RAM Read Write RAM 0 RAM 1 RAM 2 RAM 3 RAM 3 RAM 5 RAM 5 RAM 6 RAM 1 RAM 5 RAM 6 RAM 1 RAM 5 RAM 6 RAM 1 RA | rest read writ pplement 16-bit data, so 255 is cossor both have double-dabble he processor. le, the first thing you should do he "RESET pin. This primes the component labelled "Program" to Contents. There, you can di | read 24 _ A writ A_ 24 read 24 _ A writ A_ 24 read 24 _ A writ I A_ 24 writ I A_ 24 writ I A_ 25 writ I A_ 26 writ I A_ 26 read 26 _ A writ I A_ 26 | c | 0000 | 1 0 1 | | 0 0 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 | 0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 1 1 | 0 1 1 0 0 0 0 0 1 1 1 1 1 1 1 1 1 1 1 | 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 | | Read Target Address | Target Address | eg. | read((ramNo.) idx to A writ((ramNo.) A to idx | read((ramNo.) _ A B read (idx in address A) to (address B) writi(ramNo.) A B write (value in address A) to (dx in address B) |
| return RAM Read Write RAM 0 RAM 1 RAM 2 RAM 2 RAM 3 RAM 5 RAM 5 RAM 5 RAM 6 RAM 15 RAM 6 RAM 15 RAM 15 Running the Processor Unfortunately, I was unable to im The input and Output of the proc for ease of access when using th When you open the processor fli and use the pete tool to press th Next, you should find the ROM o the component and pressing "Ed program, or you can open a prog Then, you should set your input t a program, in which case you sh tick the clock, by pressing the bu | read writ | read 24 A writ A_24 read 24 A writ A_24 read 24 A writ I A_24 writ I A_24 writ I A_25 writ I A_26 writ | c c | 00000 | 1 0 1 | | 0 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 | 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 | 0 1 1 0 0 0 0 1 1 1 1 1 1 1 1 1 1 1 1 1 | 0 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 | | Read Target Address | Target Address | eg. | read((ramNo.) idx to A writ((ramNo.) A to idx | read((ramNo.) _ A B — read (dx in address A) to (address B) writt((ramNo.) A B _ — write (value in address A) to (dx in address B) |
| return RAM Read Write RAM 0 RAM 1 RAM 2 RAM 2 RAM 3 RAM 4 RAM 5 RAM 5 RAM 6 RAM 15 RAM 6 RAM 15 Running the Processor Unfortunately, I was unable to im The Input and Output of the processor floor ease of access when using th When you open the processor floor ease of access when using th When you open the processor floor ease of access when using th When you open the processor floor ease of access when using th When you open the processor floor ease of access when using th When you open the processor floor ease of access when using th When you open the processor floor ease of access when using th Then, you should find the RCM of the component and pressing "Edprogram, or you can open a prog Then, you should set your input a program, in which case you i | read writ | read 24 A writ A_24 read 24 A writ A_24 read 24 A writ I A_24 writ I A_24 writ I A_25 writ I A_26 writ | c c | 00000 | 1 0 1 | | 0 0 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 | 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 1 | 0 1 1 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 1 1 | 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 | | Read Target Address | Target Address | eg. | read((ramNo.) ldx to A writ((ramNo.) A to dx | read((ramNo.) _ A B read (idx in address A) to (address B) writi(ramNo.) A B write (value in address A) to (dx in address B) |
| return RAM Read Write RAM 0 RAM 1 RAM 2 RAM 3 RAM 3 RAM 5 RAM 6 RAM 5 RAM 6 RAM 1 RAM 5 RAM 6 RAM 1 RAM 5 RAM 6 RAM 6 RAM 1 RAM 6 RAM 6 RAM 6 RAM 6 RAM 1 RAM | read writ | read 24 _A writ A_ 24 read 24 _A writ A_ 24 readil 24 _A writ A_ 24 writ C to A_ writ A_ 24 writ C to A_ writ A_ 24 writ C to A_ | · · · | 00000 | 1 0 1 | | 0 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 | 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 | 0 1 1 0 0 0 0 0 1 1 1 1 1 0 0 0 0 0 1 | 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 | | Read Target Address | Target Address | eg. | read((ramNo.) idx to A writ((ramNo.) A to idx | read((ramNo.) _ A B — read (dx in address A) to (address B) writt((ramNo.) A B _ — write (value in address A) to (dx in address B) |
| return RAM Read Write RAM 0 RAM 1 RAM 2 RAM 2 RAM 3 RAM 5 RAM 5 RAM 5 RAM 6 RAM 1 RAM 15 RAM 15 Running the Processor Unfortunately, I was unable to im The input and Output of the proc for ease of access when using th When you open the processor fli and use the pete tool to press th Next, you should find the ROM o the component and pressing "Ed program, or you can open a prog Then, you should set your input t a program, in which case you sh tick the clock, by pressing the bu Finally, go to the "Simulate" tab it pulsing the clock. The default fer "Simulate" tab it pulsing the clock. The default fer "Simulate" tab it pulsing the clock. The default fer "Simulate" tab it pulsing the clock. The default fer "Simulate" tab it pulsing the clock. The default fer "Simulate" tab it pulsing the clock. The default fer "Simulate" tab it pulsing the clock. The default fer "Simulate" tab it pulsing the clock. The default fer simulate "tab it pulsing the clock. The default fer simulate" tab it pulsing the clock. The default fer simulate "tab it pulsing the clock. The default fer simulate "tab it pulsing the clock. The default fer simulate "tab it pulsing the clock. The default fer simulate "tab it pulsing the clock. The default fer simulate "tab it pulsing the clock. The default fer simulate "tab it pulsing the clock. The default fer simulate "tab it pulsing the clock. The default fer simulate "tab it pulsing the clock. The default fer simulate "tab it pulsing the clock. The default fer simulate "tab it pulsing the clock. The clock th | read writ | read 24 _A writ A_ 24 read 24 _A writ A_ 24 readil 24 _A writ A_ 24 writ C to A_ writ A_ 24 writ C to A_ writ A_ 24 writ C to A_ | | 00000 | 1 0 1 | 1 | 0 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 | 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 1 | 0 1 1 0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 | 0 | | Read Target Address | Target Address | eg. | read((ramNo.) idx to A wrt((ramNo.) A to idx | read((ramNo.) _ A B — read (dx in address A) to (address B) writ((ramNo.) A B _ — write (value in address A) to (dx in address B) |
| return RAM Read Write RAM 0 RAM 1 RAM 2 RAM 3 RAM 3 RAM 5 RAM 6 RAM 5 RAM 6 RAM 1 RAM 5 RAM 6 RAM 1 RAM 5 RAM 6 RAM 6 RAM 1 RAM 6 RAM 6 RAM 6 RAM 6 RAM 1 RAM | read writ | read 24 _A writ A_ 24 read 24 _A writ A_ 24 readil 24 _A writ A_ 24 writ C to A_ writ A_ 24 writ C to A_ writ A_ 24 writ C to A_ | c c | 00000 | 1 0 1 | | 0 0 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 | 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 | 0 1 1 0 0 0 0 0 1 1 1 0 0 0 0 0 1 1 1 0 0 1 | 0 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 | | Read Target Address | Target Address | eg. | read((ramNo.) idx to A writ((ramNo.) A to idx | read((ramNo.) _ A B read (idx in address A) to (address B) writt((ramNo.) A B write (value in address A) to (dx in address B) |
| return RAM Read Write RAM 0 RAM 1 RAM 2 RAM 2 RAM 3 RAM 5 RAM 5 RAM 5 RAM 6 RAM 15 RAM 6 RAM 15 Running the Processor Unfortunately, I was unable to im The input and Output of the proc for ease of access when using th When you open the processor file and use the peke tool to press th Next, you should find the ROM o the component and pressing "Ed program, or you can open a prog Then, you should set your input t a program, in which case you sh tick the clock, by pressing the bu Finally, go to the "Simulate" tab it pulsing the clock. The default fer Simulate' Tae Insulate' Auto-Tick Frequency in | read writ read read writ read rea | read 24 _A writ A_ 24 read 24 _A writ A_ 24 read 24 _A writ I A_ 24 writ I A_ 24 writ I A_ 25 writ I C I O A_ writ I A_ 25 writ I C I O A_ writ I A_ 26 writ I A_ | | 00000 | 1 0 1 1 | | 0 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 | 0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 1 1 | 0 1 1 0 0 0 0 0 1 1 0 0 0 0 1 1 1 0 0 0 0 1 | D | | Read Target Address | Target Address | eg. | read((ramNo.) idx to A writ((ramNo.) A to idx | read((ramNo.) _ A B — read (dx in address A) to (address B) witl((ramNo.) A B _ — writle (value in address A) to (dx in address B) |
| return RAM Read Write RAM 0 RAM 1 RAM 2 RAM 2 RAM 3 RAM 5 RAM 5 RAM 6 RAM 1 RAM 15 RAM 15 RAM 15 Running the Processor Unfortunately, I was unable to im The Input and Output of the proc for ease of access when using th When you open the processor fli and use the pete tool to press th Next, you should find the ROM o the component and pressing "Ed program, or you can open a prog Then, you should set your input t a program, in which case you sh tick the clock, by pressing the bu Finally, go to the "Simulate" tab it publing the clock. The default fer "Simulate" tab it publing the clock. Th | read writ read read writ read rea | read 24 _A writ A_ 24 read 24 _A writ A_ 24 read 24 _A writ I A_ 24 writ I A_ 24 writ I A_ 25 writ I C I O A_ writ I A_ 25 writ I C I O A_ writ I A_ 26 writ I A_ | | 000000000000000000000000000000000000000 | 1 0 1 | | 0 0 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 | 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 1 | 0 1 1 0 0 0 0 1 1 1 1 1 1 1 1 1 1 1 1 1 | 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 | | Read Target Address | Target Address | eg. | read((ramNo.) idx to A writ((ramNo.) A to idx | read((ramNo.) _ A B read (idx in address A) to (address B) writi(ramNo.) A B write (value in address A) to (dx in address B) |

| 1st Byte | lanta atlan | Francis | 400 | 64 | 32 | 16 | | | | 4 | 2-4 2-4- | Out Dute | All D. A. | |
|---|---------------------------------|--|-----|----|----|----|---|---|---|-----|----------|----------|-----------|--|
| 1st Byte | Instruction | Example | 128 | 64 | 32 | 16 | 8 | 4 | 2 | - 1 | 2nd Byte | 3rd Byte | 4th Byte | |
| | | | | | | | | | | | | | | |
| And Presto! | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | |
| Programming / Using the Replit pr | rogram | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | |
| Each instruction in your program v | will use 4 byte of data in the | ROM. | | | | | | | | | | | | |
| ,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,, | | | | | | | | | | | | | | |
| The first byte you type will be the | oncode | | | | | | | | | | | | | |
| The second byte you type will be it | | | | | | | | | | | | | | |
| The third byte you type will be its s | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | |
| The fourth byte you type will be the | ie address triat trie result is | sent to. | | | | | | | | | | | | |
| | | | | | | | | | | | | | | |
| When you see an "_" in an instruc | | | | | | | | | | | | | | |
| To add these bytes, you should fill | I it in with the hex number " | 20". | | | | | | | | | | | | |
| | | | | | | | | | | | | | | |
| Here is an example of what an ins | struction might look like: | | | | | | | | | | | | | |
| addij r0 24 r1> 64 0 24 1> 40 | 0001801 | * here, "jj" simply means that the second | | | | | | | | | | | | |
| † this means bitwise OR the two | values (0 OR 64 = 64) | argument is an immediate value, i.e. not an address. | | | | | | | | | | | | |
| | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | |
| Converting the binary instructions | to hey to form instructions | is hells tedique, huh? | | | | | | | | | | | | |
| That's why I've created a python p | | | | | | | | | | | | | | |
| Find it here at this link: placeholde | | (0) | | | | | | | | | | | | |
| rinu it nere at this link: placeholde | 97 | | | | | | | | | | | | | |