1st Byte	Instruction	Example	128	64	32	16	8	4	2	1	2nd Byte	3rd Byte	4th Byte		
None Immediate			0	0							Address	Address	Target Address		
2nd Immediate		addii 1 A B											Target Address		
			- 1	U							Literal	Address			
3rd Immediate	j	sublj A 1 A	0	1							Address	Literal	Target Address		
Operation															
					U	U									
Maths							0								
Add	add	add A B C					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								0	Audiess	Address	ialgerAddless		
	shl						0	0	1	1					
Shift Right	shr	shr A B C					0	1	0	0					
Negate	neg	neg A_C			0	0	0	1	0	1					
Divide	div	div A B C			0	0	0	1	1	0	Address	Address	Target Address		
Modulo	mod	mod A B C					0			1					
MODUIO	mou	MOUABC					U	1	- 1	1					
Bitwise					0	0	1								
AND	and	and A B C					1	0	0	0	Address	Address	Target Address		
OR	or	or A B C					4	0	0	4	Address	Address	Target Address		
NOT					0	0				-		Addiess	Target Address		
	not	not A_C					1	0	1	U	Address	-	Target Address		
XOR	XOL	xor A B C					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	1	0					
, and a	Andl	and no o				-		-	- 1						
Condition					0	1									
Compare against values					0		1								
Equal	io	je A B label_name				4	-	0	0	1	Address	Address	Jump Address		
	je .	je n o label_name						o .	U	1			Jump Address		
Not Equal	jne	jne A B label_name			0	1		1	0	1	Address	Address	Jump Address		
Less Than	ji .	jl A B label_name				1		0	1	0	Address	Address	Jump Address		
Less Than Or Equal To	jle	jle A B label_name			0	1		0	1	1	Address	Address	Jump Address		
Greater Than	in	jg A B label_name			0			1	1	1	Address	Address	Jump Address		
Greater Than Or Equal To	jge	jge A B label_name			0			1	1	0	Address	Address	Jump Address		
	184	like v o rener liquine			U			-1	- 1	U			Jump Address		
Space for more					0	1					Address	Address	Jump Address		
Always	jmp	jmp label_name			0	1	0	1	1	1	_	_	Jump Address		
Moving			0	0	1	0			0	0	Address		Target Address		
				U		0			0	· ·	Audiess	-	ialycthodicss		
Сору	сру	cpy A_B			1		U	U							
Move (equivalent to copy)	mov	mov A _ B					0	1							
Stack Pop	рор	pop A					1	0							
Stack Push	psh	psh A				0	1	1							
	F	F					-1	-							
Function					1	0									
									1						
call	call	call label_name				0	0	0	1	0					
		call label_name ret				0	0	0	1 1	0					
return	ret	call label_name ret					0	0	1 1 1	0					
return		call label_name ret			1	0	0	0	1 1 1	0					
return	ret	ret		1	1	0	0	0	1 1 1	0		Only 1 of these is require	d		
return		call label_name ret read 24 _ A	0	1	1 1 1	0 0 1	0	0	1 1 1	0 1	RAM Address	Read Target Address	Target Address eg.	read](ramNo.) idx to A	read((ramNo.) _ A B read (idx in address A) to (address B)
return	ret	red 24_A	0	i 0 1	1 1 1	0 0 1 1	0	0	1 1 1	0	RAM Address Address	Read Target Address	Target Address eg.	read (ramNo.) idx to A writl(ramNo.) A to idx	
return  RAM  Read  Write	ret	ret read 24 _ A writ A _ 24		1 0 1	1 1 1 1	0 0 1 1	0	0	1 1 1	0 1	RAM Address Address	Only 1 of these is require Read Target Address Read Target Address	Target Address eg.		read((ramNo.) _ A B read ((dx in address A) to (address B) writt(ramNo.) A B writte (value in address A) to (dx in address B)
return  RAM  Read  Write  RAM 0	ret	red 24 _ A writ A _ 24 read() 24 _ A		1	1 1 1 1 1	0 0 1 1 1	0	0	0 0	0 1	RAM Address Address	Read Target Address	Target Address eg.		
return  RAM  Read  Write  RAM 0  RAM 1	ret	ret read 24 _ A wrt A _ 24 read[22 _ A wrt] A _ 24		1 0 1	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	0	0	0 0	0 1 0 1 1	RAM Address Address	Read Target Address	Target Address eg.		
return  RAM  Read  Write  RAM 0  RAM 1  RAM 2	ret	red 24 _ A writ A _ 24 read() 24 _ A		1	1 1 1 1 1 1 1	0 0 1 1 1 1 1	0 0 0 0 0 0	0 0 0 0 0	0 0 1	0 1 1 1 1	RAM Address Address	Read Target Address	Target Address eg.		
return  RAM  Read  Write  RAM 0  RAM 1  RAM 2  RAM 3	ret	ret read 24 _ A wrt A _ 24 read[22 _ A wrt] A _ 24		1 0	1 1 1 1 1 1 1 1 1	0 0 1 1 1 1 1 1 1	0 0 0 0 0 0 0 0	0 0 0 0 0 0 0	0 0 1 1 1	0 1 0 1 1 1	RAM Address Address	Read Target Address	Target Address eg.		
return  RAM  Read  Write  RAM 0  RAM 1  RAM 2  RAM 3	ret	ret read 24 _ A wrt A _ 24 read[22 _ A wrt] A _ 24		1 0 1	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	0 0 0 0 0	0 0 0 0 0 0	0 0 1 1 0	0 1 1 1 1 1 0	RAM Address Address	Read Target Address	Target Address eg.		
RAM Read Write RAM 0 RAM 1 RAM 1 RAM 2 RAM 3 RAM 3	ret	ret read 24 _ A wrt A _ 24 read[22 _ A wrt] A _ 24		1 0 1	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 0 0 0 0 0 0 0	0 0 0 0 0 1 1 1	0 0 0 1 1 0 0 0 0	0 1 1 1 1 1 0 0 1 1	RAM Address Address	Read Target Address	Target Address eg.		
return  RAM  Read  Write  RAM 0  RAM 1  RAM 2  RAM 3	ret	ret read 24 _ A wrt A _ 24 read[22 _ A wrt] A _ 24		1 0 1	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 0 0 0 0 0 0 0	0 0 0 0 0 0	0 0 0 1 1 0 0 0 0 1	0 1 1 1 1 1 0 1 1 1 1	RAM Address Address	Read Target Address	Target Address eg.		
RAM Read Write RAM 0 RAM 1 RAM 1 RAM 2 RAM 3 RAM 3	ret	ret read 24 _ A wrt A _ 24 read[22 _ A wrt] A _ 24		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 0 0 0 0 0 0 0	0 0 0 0 0 0 0	0 0 1 1 0 0 0 1 1	0 1 1 1 1 1 0 0 1 1 1 1	RAM Address Address	Read Target Address	Target Address eg.		
RAM Read Write RAM 0 RAM 1 RAM 1 RAM 2 RAM 3 RAM 3	ret	ret read 24 _ A wrt A _ 24 read[22 _ A wrt] A _ 24		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	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0	0 0 0 0 0 0 0 1 1 1	1 1 1 1 0 0 0 1 1 1 1 0	0 1 0 1 1 1 1 0 1 1	RAM Address Address	Read Target Address	Target Address eg.		
RAM Read Write RAM 0 RAM 1 RAM 1 RAM 2 RAM 3 RAM 3	ret	ret read 24 _ A wrt A _ 24 read[22 _ A wrt] A _ 24		1 0 1	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 0 1 1 1 1	1 1 1 1 0 0 0 1 1 1 1 0 0	0 1 0 1 1 1 1 0 1 1 1	RAM Address Address	Read Target Address	Target Address eg.		
RAM Read Write RAM 0 RAM 1 RAM 1 RAM 2 RAM 3 RAM 3	ret	ret read 24 _ A wrt A _ 24 read[22 _ A wrt] A _ 24		i 0 1	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 0 0 0 1 1 1 1	1 1 1 0 0 0 1 1 1 0 0	0 1 0 1 1 1 0 1 1 1 0	RAM Address Address	Read Target Address	Target Address eg.		
RAM Read Write RAM 0 RAM 1 RAM 1 RAM 2 RAM 3 RAM 3	ret	ret read 24 _ A wrt A _ 24 read[22 _ A wrt] A _ 24		1 0 1	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 0 0 0 1 1 1 1	0 0 0 1 1 0 0 0 1 1	0 1 1 0 1 1 1 0 0 1 1 1 1 0	RAM Address Address	Read Target Address	Target Address eg.		
RAM Read Write RAM 0 RAM 1 RAM 1 RAM 2 RAM 3 RAM 3	ret	ret read 24 _ A wrt A _ 24 read[22 _ A wrt] A _ 24		1 0 1	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 0 0 0 1 1 1 1 1 0 0	1 1 1 1 0 0 0 1 1 1 0 0 0 1 1 1	0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1	RAM Address Address	Read Target Address	Target Address eg.		
RAM Read Write RAM 0 RAM 1 RAM 1 RAM 2 RAM 3 RAM 3	ret	ret read 24 _ A wrt A _ 24 read[22 _ A wrt] A _ 24		1 0 1	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 0 1 1 1 1 0 0	1 1 1 1 0 0 0 1 1 1 1 0 0 0 1 1 1	0 1 0 1 1 1 1 1 1 1 1 1 1 1 1 1	RAM Address Address	Read Target Address	Target Address eg.		
RAM Read Write RAM 0 RAM 1 RAM 1 RAM 2 RAM 3 RAM 3	ret	ret read 24 _ A wrt A _ 24 read[22 _ A wrt] A _ 24		1 0 1	1 1 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 0 0 0 0 0 0 1 1 1 1 0 0	1 1 1 1 0 0 1 1 1 1 0 0 0 1 1 1 1 0 0	0 1 0 1 1 1 1 0 1 1 1 1 0 1 1 1 1 1	RAM Address Address	Read Target Address	Target Address eg.		
RAM Read Write RAM 0 RAM 1 RAM 1 RAM 2 RAM 3 RAM 3	ret	ret read 24 _ A wrt A _ 24 read[22 _ A wrt] A _ 24		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 1 1 1 1 1 1 1 1 1 1 1 1	0 0 0 0 0 0 0 1 1 1 1 0 0 0	1 1 1 1 0 0 0 1 1 1 0 0 0 1 1 1 0 0	0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1	RAM Address Address	Read Target Address	Target Address eg.		
return  Read  Read  Write  RAM 0  RAM 1  RAM 1  RAM 2  RAM 3  RAM 3  RAM 5  RAM 5  RAM 5  RAM 6	ret	ret read 24 _ A wrt A _ 24 read[22 _ A wrt] A _ 24		1 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 1 1 1 1 1 1 1 1 1 1 1	0 0 0 0 0 0 0 0 1 1 1 1 0 0 0 0	1 1 1 1 1 1 0 0 0 1 1 1 0 0 0 1 1 1 0 0 0 1 1 1 0	0 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1	RAM Address Address	Read Target Address	Target Address eg.		
return  Read  Read  Write  RAM 0  RAM 1  RAM 1  RAM 2  RAM 3  RAM 3  RAM 5  RAM 5  RAM 5  RAM 6	ret	ret read 24 _ A wrt A _ 24 read[22 _ A wrt] A _ 24		1 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 1 1 1 1 1 1 1 1 1 1 1	0 0 0 0 0 0 0 0 1 1 1 1 1 0 0 0	1 1 1 1 1 1 0 0 0 1 1 1 0 0 0 1 1 1 0 0 0 1 1 1 0 0 0 1 1 1 0 0 0 1 1 1 0 0 0 1 1 1 0 0 0 1 1 1 0 0 0 1 1 1 0 0 0 1 1 1 1 1 0 0 0 1	0 1 1 1 1 1 1 1 1 1 1 1 1	RAM Address Address	Read Target Address	Target Address eg.		
RAM Read Write RAM 0 RAM 1 RAM 1 RAM 2 RAM 3 RAM 3	ret	ret read 24 _ A wrt A _ 24 read[22 _ A wrt] A _ 24		1 0 1	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 1 1 1 1 1 1 1 1 1 1 1	0 0 0 0 0 0 0 1 1 1 1 1 0 0 0 0	1 1 1 1 0 0 0 1 1 1 0 0 0 1 1 1 0 0 0 1 1 1 0 0 0 0 1 1 1 0	0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1	RAM Address Address	Read Target Address	Target Address eg.		
Return Read Read Write RAM 0 RAM 1 RAM 1 RAM 2 RAM 2 RAM 3 RAM 4 RAM 5 RAM 6 RAM 6 RAM 6 RAM 6 RAM 6 RAM 7	ret	ret read 24 _ A wrt A _ 24 read[22 _ A wrt] A _ 24		1 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 1 1 1 1 1 1	0 0 0 0 0 0 0 1 1 1 1 1 0 0 0 0	1 1 1 1 1 1 0 0 1 1 1 0 0 1 1 1 0 0 1 1 1 1 0 0 1 1 1 1 0 0 1 1 1 1 0 0 1 1 1 1 0 0 0 1	0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1	RAM Address Address	Read Target Address	Target Address eg.		
return  Read  Read  Write  RAM 0  RAM 1  RAM 1  RAM 2  RAM 3  RAM 3  RAM 5  RAM 5  RAM 5  RAM 6	ret	ret read 24 _ A wrt A _ 24 read[24 _ A wrt I A _ 24		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 1 1 1 1 1 1 1 1 1 1	0 0 0 0 0 0 0 1 1 1 1 1 0 0 0 0 0	1 1 1 1 0 0 0 1 1 1 0 0 0 1 1 1 0 0 0 1 1 1 0 0 0 0 1 1 1 0 0 0 0 1 1 1 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	RAM Address Address	Read Target Address	Target Address eg.		
Return  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 7  RAM 7  RAM 7  RAM 7  RAM 7  RAM 8  RAM 8  RAM 9  RAM 15	red read writ	ret read 24 _ A writ A_ 24 readig 24 _ A writ A_ 24 writ I A_ 24 writ I A_ 24 writ I A_ 24		1 0 1	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 1 1 1 1 1 1	0 0 0 0 0 0 0 1 1 1 1 1 0 0 0 0 0	1 1 1 1 1 1 0 0 1 1 1 0 0 1 1 1 0 0 1 1 1 0 0 0 1 1 1 1 0 0 0 0 0 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	RAM Address Address	Read Target Address	Target Address eg.		
Return  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 7  RAM 7  RAM 7  RAM 7  RAM 7  RAM 8  RAM 8  RAM 9  RAM 15	red read writ	ret read 24 _ A writ A_ 24 readig 24 _ A writ A_ 24 writ I A_ 24 writ I A_ 24 writ I A_ 24		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 1 1 1 1 1 1 1 1 1 1	0 0 0 0 0 0 0 1 1 1 1 1 0 0 0 0	1 1 1 1 1 0 0 0 1 1 1 0 0 0 1 1 1 0 0 1 1 0 0 1 1 0 0 1 1 0 0 0 1 1 0 0 0 0 1 1 0 0 0 0 1 1 0 0 0 1 1 0 0 0 0 1 1 1 0 0 0 0 1 1 1 1 1 0 0 0 0 1	0 1 1 1 1 1 0 0 1 1 1 1 1 1 1 1 1 1 1 1	RAM Address Address	Read Target Address	Target Address eg.		
Return Read Read Write RAM 0 RAM 1 RAM 1 RAM 2 RAM 2 RAM 3 RAM 4 RAM 5 RAM 6 RAM 6 RAM 6 RAM 6 RAM 6 RAM 7	red read writ	ret read 24 _ A writ A_ 24 readig 24 _ A writ A_ 24 writ I A_ 24 writ I A_ 24 writ I A_ 24		1 0 1	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 1 1 1 1 1	0 0 0 0 0 0 0 1 1 1 1 1 0 0 0 0	1 1 1 1 1 0 0 0 1 1 1 1 0 0 0 1 1 1 1 0 0 0 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	RAM Address Address	Read Target Address	Target Address eg.		
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 7  RAM 7  RAM 7  RAM 8  RAM 8  RAM 9  RAM 15	ret read writ  mplement 16-bit data, so 256 i	ret read 24 _ A writ A_ 24 readig 24 _ A writ A_ 24 writ (C to A		1 0 1	1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1	O O O O O O O O O O O O O O O O O O O	0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1	0 0 0 0 0 0 0 1 1 1 1 1 0 0 0 0	1 1 1 1 1 1 0 0 1 1 1 0 0 0 1 1 1 0 0 0 1 1 1 0 0 0 1 1 1 0 0 0 0 1 1 1 0	0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1	RAM Address Address	Read Target Address	Target Address eg.		
Return Read Write RAM 0 RAM 0 RAM 1 RAM 2 RAM 3 RAM 5 RAM 5 RAM 5 RAM 6 RAM 6 RAM 15 RAM 15 Raming the Processor Urifortunately, I was unable to imp	ret read writ  mplement 16-bit data, so 255i	ret read 24 _ A writ A_ 24 readig 24 _ A writ A_ 24 writ (C to A		1 0 1	1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1	O O O O O O O O O O O O O O O O O O O	0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1	0 0 0 0 0 0 0 1 1 1 1 1 0 0 0 0 1 1 1 1	1 1 1 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	RAM Address Address	Read Target Address	Target Address eg.		
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 7  RAM 7  RAM 7  RAM 8  RAM 8  RAM 9  RAM 15	ret read writ  mplement 16-bit data, so 255i	ret read 24 _ A writ A_ 24 readig 24 _ A writ A_ 24 writ (C to A		1 0 1	1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1	O O O O O O O O O O O O O O O O O O O	0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1	0 0 0 0 0 0 1 1 1 1 0 0 0 0 0 0 1 1 1 1	1 1 1 1 1 1 0 0 1 1 1 0 0 1 1 1 0 0 1 1 1 0 0 1 1 1 1 0 0 1 1 1 1 1 0 0 0 1	0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1	RAM Address Address	Read Target Address	Target Address eg.		
Return RAM Reed Write RAM 0 RAM 1 RAM 2 RAM 3 RAM 3 RAM 4 RAM 5 RAM 6 RAM 6 RAM 6 RAM 15 Running the Processor Urifortunately, I was unable to imp	ret read writ  mplement 16-bit data, so 255: cessor both have double-dabl the processor.	ret  read 24 _ A  wrd A_ 24  read) 24 _ A  wrtt A_ 24  wrtt A_ 24  wrtt C to A_  sthe max right now - sorry.		1 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 1 1 1 1 1 1 1 1 1 1	0 0 0 0 0 0 0 1 1 1 1 1 0 0 0 0 0 1 1 1 1 1 1 1	1 1 1 1 1 1 0 0 0 1 1 1 1 0 0 0 1 1 1 0 0 1 1 1 0 0 0 1 1 1 0 0 0 0 1 1 0 0 0 1 1 0 0 0 1 1 1 0 0 0 1 1 1 1 0 0 0 1 1 1 1 0 0 0 1	0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1	RAM Address Address	Read Target Address	Target Address eg.		
Return RAM Reed Write RAM 0 RAM 1 RAM 2 RAM 3 RAM 3 RAM 4 RAM 5 RAM 6 RAM 6 RAM 6 RAM 15 Running the Processor Urifortunately, I was unable to imp	ret read writ  mplement 16-bit data, so 255: cessor both have double-dabl the processor.	ret  read 24 _ A  wrd A_ 24  read) 24 _ A  wrtt A_ 24  wrtt A_ 24  wrtt C to A_  sthe max right now - sorry.		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 1 1 1	0 0 0 0 0 0 0 1 1 1 1 0 0 0 0 0 0 0 1	1 1 1 1 1 1 0 0 1 1 1 0 0 1 1 1 0 0 1 1 1 1 0 0 0 1	0 1 1 1 1 1 0 0 1 1 1 1 1 1 1 1 1 1 1 1	RAM Address Address	Read Target Address	Target Address eg.		
Return  Read  Write  RAM 0  RAM 1  RAM 2  RAM 2  RAM 3  RAM 4  RAM 5  RAM 5  RAM 5  RAM 6  Unifortunately, I was unable to imp  The Input and Output of the processor  The Input and Output of the processor for ease of aces when using the pro	ret read writ  mplement 16-bit data, so 256 cessor both have double-dabl the processor.	red 24 _A wxt A_ 24 read(0) 24 _A wxt I A_ 24 wxt I A_ 24 wxt I A_ 24 wxt I O A_  st the max right now - sorry.  ble implementation		1 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 1 1 1 1 1 1 1 1 1 1	0 0 0 0 0 0 0 1 1 1 1 0 0 0 0 0 1 1 1 1	1 1 1 1 1 0 0 0 1 1 1 0 0 0 1 1 1 0 0 0 1 1 1 0 0 1 1 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	RAM Address Address	Read Target Address	Target Address eg.		
Return RAM Reed Write RAM 0 RAM 1 RAM 2 RAM 3 RAM 3 RAM 4 RAM 5 RAM 6 RAM 15 Running the Processor Unfortunately, I was unable to imp	ret read writ  mplement 16-bit data, so 256 cessor both have double-dabl the processor.	red 24 _A wxt A_ 24 read(0) 24 _A wxt I A_ 24 wxt I A_ 24 wxt I A_ 24 wxt I O A_  st the max right now - sorry.  ble implementation		1 0 1	1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1	O O O O O O O O O O O O O O O O O O O	0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1	0 0 0 0 0 0 1 1 1 1 1 0 0 0 0 0 0 1	1 1 1 1 0 0 0 1 1 1 1 0 0 0 1 1 1 1 0 0 0 1 1 1 1 0 0 0 1 1 1 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	RAM Address Address	Read Target Address	Target Address eg.		
RAM Read Write RAM 0 RAM 1 RAM 2 RAM 2 RAM 3 RAM 4 RAM 5 RAM 5 RAM 5 RAM 5 RAM 5 RAM 6 RAM 5 RAM 6 RAM 7 RAM 9 RAM 15 Ram 10 Ram	ret read writ  Implement 16-bit data, so 255 cessor both have double-dabt the processor.  Ile, the first thing you should o	ret read 24 _A wrt A_ 24 read() 24 _A wrt() 1A_ 24 wrt() 10 A  st the max right now - sorry.  lole implementation  lo is find the i/o area the RAM banks.		1 0 1	1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1	O O O O O O O O O O O O O O O O O O O	0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 1 1	0 0 0 0 0 0 1 1 1 1 1 1 0 0 0 0 0 0 1	1 1 1 1 1 1 0 0 1 1 1 0 0 0 1 1 1 1 0 0 0 1 1 1 0 0 1 1 1 0 0 0 1 1 1 1 0 0 1	0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1	RAM Address Address	Read Target Address	Target Address eg.		
Return Read Write RAM 0 RAM 1 RAM 2 RAM 3 RAM 3 RAM 5 RAM 5 RAM 5 RAM 6 RAM 6 RAM 15 RAM 15 Running the Processor Unifortunately, I was unable to imp	ret read writ  mplement 16-bit data, so 256 cessor both have double-dabl the processor. lie, the first thing you should of the "RESET" pin. This primes component labelled "Program	ret  read 24 _A writ A_ 24 read) 24 _A writ I A_ 24 writ I A_ 24 writ C io A_  s the max right now - sorry.  ble implementation o is find the i/o area the RAM banks.  and edit the contents right-clicking		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	O O O O O O O O O O O O O O O O O O O	0 0 0 0 0 0 0 1 1 1 1 0 0 0 0 0 0 1 1 1 1 1 1	1 1 1 1 1 1 0 0 0 1 1 1 0 0 0 1 1 1 0 0 0 1 1 1 0 0 0 1 1 1 0 0 0 1 1 1 1 0 0 1	0 1 1 1 1 1 0 0 1 1 1 1 1 1 1 1 1 1 1 1	RAM Address Address	Read Target Address	Target Address eg.		
RAM Read Write RAM 0 RAM 1 RAM 2 RAM 1 RAM 2 RAM 3 RAM 4 RAM 5 RAM 5 RAM 5 RAM 5 RAM 6 RAM	ret read writ  mplement 16-bit data, so 255: cessor both have double-dabl the processor. ite, the first thing you should o the "RESET" pin. This primes component labelled "Progs component labelled "Progs component labelled". There, you can	ret  read 24 _A writ A_ 24 read) 24 _A writ I A_ 24 writ I A_ 24 writ C io A_  s the max right now - sorry.  ble implementation o is find the i/o area the RAM banks.  and edit the contents right-clicking		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 1 1 1 1	0 0 0 0 0 0 0 1 1 1 1 1 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 1 1	1 1 1 1 1 1 0 0 1 1 1 0 0 0 1 1 1 0 0 0 1 1 1 1 0 0 1 1 1 1 0 0 1	0 1 1 1 1 1 1 0 0 1 1 1 1 1 1 1 1 1 1 1	RAM Address Address	Read Target Address	Target Address eg.		
RAM Read Write RAM 0 RAM 1 RAM 2 RAM 1 RAM 2 RAM 3 RAM 4 RAM 5 RAM 5 RAM 5 RAM 5 RAM 6 RAM	ret read writ  mplement 16-bit data, so 255: cessor both have double-dabl the processor. ite, the first thing you should o the "RESET" pin. This primes component labelled "Program component labelled "Program component labelled". There, you can	ret  read 24 _A writ A_ 24 read) 24 _A writ I A_ 24 writ I A_ 24 writ C io A_  s the max right now - sorry.  ble implementation o is find the i/o area the RAM banks.  and edit the contents right-clicking		1 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 1 1 1 1 1 1 1 1 1 1	0 0 0 0 0 0 0 1 1 1 1 1 0 0 0 0 0 1 1 1 1 1 1 1	1 1 1 1 1 1 0 0 0 1 1 1 1 0 0 0 1 1 1 1	0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1	RAM Address Address	Read Target Address	Target Address eg.		
return  RAM  Read  Write  RAM 0  RAM 1  RAM 2  RAM 3  RAM 3  RAM 5  RAM 5  RAM 5  RAM 15  RAM 15  Running the Processor  Unfortunately, I was unable to imp  The Input and Output of the proof for ease of access when using the value of the processor field of the proof of the proo	ret read writ  mplement 16-bit data, so 255: cessor both have double-dabl the processor. ite, the first thing you should o the "RESET" pin. This primes component labelled "Program component labelled "Program component labelled". There, you can	ret  read 24 _A writ A_ 24 read) 24 _A writ I A_ 24 writ I A_ 24 writ C io A_  s the max right now - sorry.  ble implementation o is find the i/o area the RAM banks.  and edit the contents right-clicking		1	1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1	O O O O O O O O O O O O O O O O O O O	0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1	0 0 0 0 0 0 1 1 1 1 1 0 0 0 0 0 0 1 1 1 1 1 1 1	1 1 1 1 1 0 0 0 1 1 1 1 0 0 0 1 1 1 1 0 0 0 1 1 1 1 0 0 0 1	0 1 1 1 1 1 0 0 1 1 1 1 1 1 1 1 1 1 1 1	RAM Address Address	Read Target Address	Target Address eg.		
RAM Read Write RAM 0 RAM 1 RAM 2 RAM 1 RAM 2 RAM 3 RAM 4 RAM 5 RAM 5 RAM 5 RAM 5 RAM 6 RAM 5 RAM 6 RAM 5 RAM 6 RAM 6 RAM 7 RAM 8 RAM 6 RAM	ret read writ  mplement 16-bit data, so 255 i cessor both have double-dabl the processor. lie, the first thing you should d the "RESET" pin. This primes component labelled "Program did Contents". There, you can gram file from your PC.	red 24 A wx A _ 24 read 24 A wx A _ 24 read) 24 A wxtq I A _ 24 wxtq I b A _ 2		1 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 1 1 1 1 1 1 1 1 1 1	0 0 0 0 0 0 0 1 1 1 1 0 0 0 0 0 0 1 1 1 1 1 1 1	1 1 1 1 1 0 0 0 1 1 1 0 0 0 1 1 1 0 0 0 1 1 1 0 0 1 1 1 0 0 0 1 1	0 1 1 1 1 1 0 0 1 1 1 1 1 1 1 1 1 1 1 1	RAM Address Address	Read Target Address	Target Address eg.		
return  RAM  Read  Write  RAM 0  RAM 1  RAM 2  RAM 3  RAM 3  RAM 5  RAM 5  RAM 5  RAM 5  RAM 6  RAM 15  RAM 15  Running the Processor  Unfortunately, I was unable to imp  The input and Output of the processor for ease of access when using the  When you open the processor fit and use the poke toot to pross the  and use the poke toot to pross the  Next, you should find the ROM oc the component and pressing "Edi program, or you approach program, or you pan program.	ret  read writ  mplement 16-bit data, so 256: cessor both have double-dabl the processor.  ile, the first thing you should of the "RESET" pin. This primes component labelled "Program dit Contents". There, you can gram file from your PC.	ret  read 24 _ A  writ A_ 24  read) 24 . A  writ I A_ 24  writ I A_ 24  writ I C io A_  s the max right now - sorry.  ble implementation  to is find the i/o area the RAM banks.  and edit the contents right-clicking directly implement your own  hanged in the midst of		1 0 1	1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1	O O O O O O O O O O O O O O O O O O O	O O O O O O O O O O O O O O O O O O O	0 0 0 0 0 0 1 1 1 1 1 0 0 0 0 0 1 1 1 1	1 1 1 1 1 0 0 0 1 1 1 0 0 0 1 1 1 1 0 0 0 1 1 1 1 0 0 0 1 1 1	0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1	RAM Address Address	Read Target Address	Target Address eg.		
RAM Read Write RAM 0 RAM 1 RAM 2 RAM 1 RAM 2 RAM 3 RAM 4 RAM 5 RAM 5 RAM 5 RAM 5 RAM 6 RAM 5 RAM 6 RAM	ret  read writ  read writ  mplement 16-bit data, so 255 i cessor both have double-dabl he processor.  lie, the first thing you should of the "RESET" pin. This primes control to the state of the state	red 24 A wx A _ 24 read 24 A wx A _ 24 read) 24 A wxit I A _ 24 wxit I A _ 24 wxit I O A _ with I A _ 24 wxit I O A _ with I A _ 24 wxit I O A _ wxit I A _ 24 wxit I O A _ wxit I A _ 24 wxit I O A _ w		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 1 1 1 1 1 1 1 1 1 1	0 0 0 0 0 0 1 1 1 1 1 0 0 0 0 0 1 1 1 1	1 1 1 1 1 1 0 0 1 1 1 0 0 0 1 1 1 1 0 0 0 1 1 1 1 0 0 1 1 1 1 0 0 1	0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1	RAM Address Address	Read Target Address	Target Address eg.		
RAM Read Write RAM 0 RAM 1 RAM 2 RAM 1 RAM 2 RAM 3 RAM 4 RAM 5 RAM 5 RAM 5 RAM 5 RAM 6 RAM 5 RAM 6 RAM	ret  read writ  read writ  mplement 16-bit data, so 255 i cessor both have double-dabl he processor.  lie, the first thing you should of the "RESET" pin. This primes control to the state of the state	red 24 A wx A _ 24 read 24 A wx A _ 24 read) 24 A wxit I A _ 24 wxit I A _ 24 wxit I O A _ with I A _ 24 wxit I O A _ with I A _ 24 wxit I O A _ wxit I A _ 24 wxit I O A _ wxit I A _ 24 wxit I O A _ w		1 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	O O O O O O O O O O O O O O O O O O O	0 0 0 0 0 0 1 1 1 1 1 0 0 0 0 0 1 1 1 1	1 1 1 1 1 1 0 0 0 1 1 1 1 0 0 0 0 1 1 1 1 0 0 0 1 1 1 1 1 1 1 0 1	0 1 1 1 1 1 1 0 0 1 1 1 1 1 1 1 1 1 1 1	RAM Address Address	Read Target Address	Target Address eg.		
return  RAM  Read  Write  RAM 0  RAM 1  RAM 2  RAM 3  RAM 3  RAM 5  RAM 5  RAM 5  RAM 5  RAM 6  RAM 15  RAM 15  Running the Processor  Unfortunately, I was unable to imp  The input and Output of the processor for ease of access when using the  When you open the processor fit and use the poke toot to pross the  and use the poke toot to pross the  Next, you should find the ROM oc the component and pressing "Edi program, or you approach program, or you pan program.	ret  read writ  read writ  mplement 16-bit data, so 255 i cessor both have double-dabl he processor.  lie, the first thing you should of the "RESET" pin. This primes control to the state of the state	red 24 A wx A _ 24 read 24 A wx A _ 24 read) 24 A wxit I A _ 24 wxit I A _ 24 wxit I O A _ with I A _ 24 wxit I O A _ with I A _ 24 wxit I O A _ wxit I A _ 24 wxit I O A _ wxit I A _ 24 wxit I O A _ w		1 0 1	1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1	O O O O O O O O O O O O O O O O O O O	0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1	0 0 0 0 0 0 1 1 1 1 1 0 0 0 0 0 1 1 1 1	1 1 1 1 1 1 0 0 1 1 1 0 0 0 1 1 1 0 0 0 1 1 1 1 1 0 0 1	0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1	RAM Address Address	Read Target Address	Target Address eg.		
RAM Read Write RAM 0 RAM 1 RAM 2 RAM 2 RAM 3 RAM 3 RAM 4 RAM 5 RAM 5 RAM 5 RAM 5 RAM 6 RAM 6 RAM 15 Raming the Processor Unfortunately, I was unable to imp The Input and Output of the proce for ease of access when using the When you open the processor fle and use the peke tool to press the Next, you should find the ROM or the component and pressing "Edi program, or you can open a prog Then, you should set your input v a program, or you can open a prog	ret read writ  read writ  mplement 16-bit data, so 255: cessor both have double-dabl the processor.  lie, the first thing you should of the "RESET" pin. This primes dit Contents". There, you can gram file from your PC. value. This may need to be could NOT do the next step. F utton directly right of the midd	read 24 _ A wx 4 _ 24 read 24 _ A wx 4 _ 24 read) 24 _ A wxit 1 _ 24 wxit G to A _  site max right now - sorry.  ble implementation  lo is find the i/o area the RAM banks.  "and edit the contents right-dicking directly implement your own  hanged in the midst of tather, you should hand le button in the next step		1 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 1 1 1 1 1 1 1	0 0 0 0 0 0 0 1 1 1 1 1 0 0 0 0 0 1 1 1 1 1 1 1	1 1 1 1 1 1 0 0 0 1 1 1 1 0 0 0 1 1 1 0 0 1 1 1 0 0 1 1 1 1 0 0 1 1 1 1 0 0 1	0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1	RAM Address Address	Read Target Address	Target Address eg.		
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 1 RAM 1 RAM 6 RAM 1 RAM 6 RAM 15 Running the Processor Unfortunately, I was unable to imp The Input and Output of the processor for ease of access when using th When you open the processor fit and use the poke too to press th And use the poke too to press th Next, you should find the ROM oc the component and pressing "Edi program, or you can open a prog Then, you anopen a prog Then, you should set your input v a program, in which case you she tok the clock, by pressing the but Finally, go to the "Simulate" tab in	ret  read writ  read writ  mplement 16-bit data, so 255: cessor both have double-dabl the processor.  lie, the first thing you should of the "RESET" pin. This primes component labelled "Program dit Contents". There, you can gram file from your PC.  value. This may need to be c hould NOT do the next step. F. value this may need to be c hould NOT do the next step. F.	red read 24 _A writ A_ 24 read) 23 _A writ 1 A_ 24 writ(C io A_  s the max right now - sorry.  ble implementation  io is find the liv area the RAM banks.  "and edit the contents right-clicking directly implement your own  hanged in the midst of tather, you should hand be tutton in the met step  iddde button to begin		1 0 1	1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1	O O O O O O O O O O O O O O O O O O O	0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1	0 0 0 0 0 0 1 1 1 1 1 0 0 0 0 0 1 1 1 1	1 1 1 1 1 0 0 0 1 1 1 1 0 0 0 1 1 1 1 0 0 0 1 1 1 1 1 1 1 1 1 0 0 0 1	0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1	RAM Address Address	Read Target Address	Target Address eg.		
RAM Read Write RAM0 0 RAM1 1 RAM2 2 RAM3 2 RAM4 3 RAM4 5 RAM5 5 RAM5 6 RAM5 6 RAM5 6 RAM6 6 RAM6 6 RAM6 6 RAM6 6 RAM6 7 R	ret read writ read writ read writ reserver reser	red read 24 _A writ A_ 24 read) 23 _A writ 1 A_ 24 writ(C io A_  s the max right now - sorry.  ble implementation  io is find the liv area the RAM banks.  "and edit the contents right-clicking directly implement your own  hanged in the midst of tather, you should hand be tutton in the met step  iddde button to begin		1 0 0 1	1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1	O O O O O O O O O O O O O O O O O O O	0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 1 1	0 0 0 0 0 0 0 1 1 1 1 1 0 0 0 0 0 0 1 1 1 1 1 1 1	1 1 1 1 1 0 0 0 1 1 1 0 0 0 1 1 1 0 0 0 1 1 1 1 0 0 1 1 1 1 1	0 1 1 1 1 1 1 0 0 1 1 1 1 1 1 1 1 1 1 1	RAM Address Address	Read Target Address	Target Address eg.		
return  RAM  Read  Write  RAM 0  RAM 1  RAM 2  RAM 3  RAM 3  RAM 5  RAM 5  RAM 6  RAM 15  RAM 15  RAM 15  RAM 15  RAM 16  RAM 15  Running the Processor  Unfortunately, I was unable to imp  The linput and Output of the procet for ease of access when using the  When you open the processor fit and use the poke too to press the and use the poke too to press the Next, you should find the ROM oc the component and pressing "Edi program, or you can open a progr  Then, you should set your input v a program, in which case you an high  Kith Red Coke, by pressing the but Finally, go to the "Simulate" tab in  Finally, go to the "Simulate" tab in	ret read writ read writ read writ reserver reser	red read 24 _A writ A_ 24 read) 23 _A writ 1 A_ 24 writ(C io A_  s the max right now - sorry.  ble implementation  io is find the liv area the RAM banks.  "and edit the contents right-clicking directly implement your own  hanged in the midst of tather, you should hand be tutton in the met step  iddde button to begin		1 0 1	1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1	O O O O O O O O O O O O O O O O O O O	O O O O O O O O O O O O O O O O O O O	0 0 0 0 0 0 1 1 1 1 1 0 0 0 0 0 1 1 1 1	1 1 1 1 1 0 0 0 1 1 1 1 0 0 0 1 1 1 1 0 0 0 1	0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1	RAM Address Address	Read Target Address	Target Address eg.		
RAM Read Write RAM0 0 RAM1 1 RAM2 2 RAM3 2 RAM4 3 RAM4 5 RAM5 5 RAM5 6 RAM5 6 RAM5 6 RAM6 6 RAM6 6 RAM6 6 RAM6 6 RAM6 7 R	ret read writ read writ read writ reserver reser	red read 24 _A writ A_ 24 read) 23 _A writ 1 A_ 24 writ(C io A_  s the max right now - sorry.  ble implementation  io is find the liv area the RAM banks.  "and edit the contents right-clicking directly implement your own  hanged in the midst of tather, you should hand be tutton in the met step  iddde button to begin		1 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 1 1 1 1 1 1	0 0 0 0 0 0 1 1 1 1 1 0 0 0 0 0 1 1 1 1	1 1 1 1 1 1 0 0 0 1 1 1 0 0 0 1 1 1 1 0 0 0 1 1 1 1 1 1 1 1 1 0 1	0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1	RAM Address Address	Read Target Address	Target Address eg.		
return  RAM  Read  Write  RAM 0  RAM 1  RAM 2  RAM 2  RAM 3  RAM 4  RAM 5  RAM 5  RAM 6  Write  RAM 15  RAM 6  RAM	ret read writ read writ read writ reserver reser	read 24 _ A wxt A 24 read 24 _ A wxt A 24 read) 24 _ A wxt I, A 24 wxt I, C 24 wxt I, C 24 wxt I, C 26 wxt I, C		1 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	O O O O O O O O O O O O O O O O O O O	0 0 0 0 0 0 1 1 1 1 1 0 0 0 0 1 1 1 1 1	1 1 1 1 1 1 0 0 0 1 1 1 1 0 0 0 1 1 1 1	0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1	RAM Address Address	Read Target Address	Target Address eg.		
return  RAM  Raed  Write  RAM 0  RAM 1  RAM 2  RAM 2  RAM 3  RAM 4  RAM 5  RAM 5  RAM 6  RAM 6  RAM 15  RAM 15  Ranning the Processor  Unfortunately, I was unable to imp  The Input and Output of the proce for ease of access when using th  When you open the processor file and use the poke tool to press th  Next, you should find the ROM or the component and pressing "Edi program, or you can open a prog  Then, you should set your input v  a program, or you can open a prog  Then, you should set your input v  a program, in which case you sho tick the clock, by pressing the but  Finally, go to the "Simulate" talk in  pulsing the clock. The default calk in	ret read writ read writ read writ reserver reser	read 24 _ A wxt A 24 read 24 _ A wxt A 24 read) 24 _ A wxt I, A 24 wxt I, C 24 wxt I, C 24 wxt I, C 26 wxt I, C		1 0 1	1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1	O O O O O O O O O O O O O O O O O O O	0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1	0 0 0 0 0 0 1 1 1 1 1 0 0 0 0 0 1 1 1 1	1 1 1 1 1 0 0 0 1 1 1 1 0 0 0 1 1 1 1 0 0 0 1	0 1 1 1 1 1 0 0 1 1 1 1 1 1 1 1 1 1 1 1	RAM Address Address	Read Target Address	Target Address eg.		

1st Byte	Instruction	Example	128	3	64	32	16	8	4	2	1	2nd Byte	3rd Byte	4th Byte	
nd Presto!															
rogramming / Using the F	Replit program														
ach instruction in your pr	ogram will use 4 byte of data in the	ROM.													
he first byte you type will															
he second byte you type															
he third byte you type wil	be its second argument.														
he fourth byte you type w	ill be the address that the result is	sent to.													
	instruction's example, it means th														
o add these bytes, you sh	ould fill it in with the hex number ".	20".													
	t an instruction might look like:														
ıddij r0 24 r1> 64 0 24		* here, " j" 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 addre	988.												
Converting the binary instr	uctions to hex to form instructions	is hella tedious, huh?													
That's why I've created a p	ython program so you don't have t	o!													
Find it here at this link: http	s://replit.com/@LucasChambers/L	EG-Code-Generation?v=1													
The program follows the sa	ame syntax as described in the exa	amples in this guide.													
o type comments, type in	a # at the end of an instruction. E	.g. mov i 27 to r1 # move 27 to r1													
ou can also just leave en	ire lines as comments with a '#' at	the start of the line.													
All opcodes execute in ord	er from left to right, i.e (add a b c)	= a + b> c													
Some special cases:															
o denote what RAM bank	you are addressing, type " after re	ead/writ and then the address of the bank													
	address, etc. to signify the bank a														
		er arg 1, 2 or both are immediates, respectively.													
You can use the keyword '	to' as a stand-in for ' 'to improve r	eadability - both function as explicit empties.													
	your typed code, type 'end' on a n	our line													

	Туре			256	128	64	32	16	8	4	2	1	
	Register		0	0	0	0	0						
	IO		1	0	0	1	0						These can be
	Stack		2	0	1	0	0						accessed like any
	Program Counter		3	0	1	1	0						other register
	Explicitly Empty "_"						1						
	Console Offset			1	0	0	0						
Register	Register	AH	0		0	0	0	2	2	X	X	X	
	rogiotoi	7 411	0		0	0	· ·		:				
Ю	Input	io	1		0	1	0	?	?	0	0	0	
	Output	io	1		0	1	0	?	?	0	0	0	
Stack	Stack	stack	2		1	0	0	0	0	0	0	0	
	Function Stack	Not ad	2		1	0	0	0	0	0	0	1	
PC	Program Counter	pc	3		1	1	0	0	0	0	0	0	
RAM	RAM Banks	ram											