PHÖNIX - 1

instruction set

MODE	NAME		word 1 OP-Code	word 1 DEST./SOURCE	word 2 SOURCE/val	SIZE	CYCLES	description
move	mov	0x10	0b00 01 0000	reg | reg	-	2 Byte	?	move reg to reg
	imov	0x11	0b00 01 0001	reg | -	immediate	4 Byte	?	move immediate to reg
	write	0x12	0b00 01 0010	reg | reg	-	2 Byte	?	write to RAM
	iwrite	0x13	0b00 01 0011	reg | -	immediate	4 Byte	?	write immediate to RAM
	load	0x14	0b00 01 0100	reg | -	reg | -	4 Byte	?	load from RAM to reg
	push	0x15	0b00 01 0101	- | reg	-	2 Byte	?	push reg to stack
	pop	0x16	0b00 01 0110	reg | -	-	2 Byte	?	pop from stack to reg
	call	0x17	0b00 01 0111	-	-	2 Byte	?	
	return	0x18	0b00 01 1000	-	-	2 Byte	?	
	-	0x19	0b00 01 1001	-	-	-	-	-
	-	0x1a	0b00 01 1010	-	-	-	-	-
	-	0x1b	0b00 01 1011	-	-	-	-	-
	-	0x1c	0b00 01 1100	-	-	-	-	-
	-	0x1d	0b00 01 1101	-	-	-	-	-
	jmp	0x1e	0b00 01 1110	reg | -	-	2 Byte	?	jump to address
	jmpI	0x1f	0b00 01 1111	-	immediate	4 Byte	?	jump to immediate

MODE	NAME		word 1 OP-Code	word 1 DEST./SOURCE	word 2 SOURCE/val	SIZE	CYCLES	description
arithmetic logic	add	0x10	0b00 01 0000	reg | reg	reg | -	4 Byte	?	A + B
	adc	0x11	0b00 01 0001	reg | reg	reg | -	4 Byte	?	A + B + C
	sub	0x12	0b00 01 0010	reg | reg	reg | -	4 Byte	?	A - B
	sbb	0x13	0b00 01 0011	reg | reg	reg | -	4 Byte	?	A - B - C
	mul	0x14	0b00 01 0100	reg | reg	reg | -	4 Byte	?	A * B
	div	0x15	0b00 01 0101	reg | reg	reg | -	4 Byte	?	A / B
	inc	0x16	0b00 01 0110	reg | reg	-	2 Byte	?	A + 1
	dec	0x17	0b00 01 0111	reg | reg	-	2 Byte	?	A - 1
	and	0x18	0b00 01 1000	reg | reg	reg | -	4 Byte	?	A & B
	or	0x19	0b00 01 1001	reg | reg	reg | -	4 Byte	?	A | B
	xor	0x1a	0b00 01 1010	reg | reg	reg | -	4 Byte	?	A ^ B
	not	0x1b	0b00 01 1011	reg | reg	-	2 Byte	?	~B
	neg	0x1c	0b00 01 1100	reg | reg	-	2 Byte	?	~B + 1
	ls	0x1d	0b00 01 1101	reg | reg	-	2 Byte	?	B << 1
	rsl	0x1e	0b00 01 1110	reg | reg	-	2 Byte	?	B >>> 1
	rsa	0x1f	0b00 01 1111	reg | reg	-	2 Byte	?	B >> 1

MODE	NAME		word 1 OP-Code	word 1 DEST./SOURCE	word 2 SOURCE/val	SIZE	CYCLES	description
condition	jz	0x10	0b00 01 0000	- | reg	-	2 Byte	?	jump if zero
	jnz	0x11	0b00 01 0001	- | reg	-	2 Byte	?	jump if not zero
	js	0x12	0b00 01 0010	- | reg	-	2 Byte	?	jump if signed
	jns	0x13	0b00 01 0011	- | reg	-	2 Byte	?	jump if not signed
	jc	0x14	0b00 01 0100	- | reg	-	2 Byte	?	jump if carry
	jnc	0x15	0b00 01 0101	- | reg	-	2 Byte	?	jump if not carry
	jo	0x16	0b00 01 0110	- | reg	-	2 Byte	?	jump if overflow
	jno	0x17	0b00 01 0111	- | reg	-	2 Byte	?	jump if not overflow
	jp	0x18	0b00 01 1000	- | reg	-	2 Byte	?	jump if even
	jnp	0x19	0b00 01 1001	- | reg	-	2 Byte	?	jump if odd
	je	0x1a	0b00 01 1010	- | reg	-	2 Byte	?	jump if equal
	jne	0x1b	0b00 01 1011	- | reg	-	2 Byte	?	jump if not equal
	jg	0x1c	0b00 01 1100	- | reg	-	2 Byte	?	jump if greater (unsigned)
	jgs	0x1d	0b00 01 1101	- | reg	-	2 Byte	?	jump if greater (signed)
	jl	0x1e	0b00 01 1110	- | reg	-	2 Byte	?	jump if less (unsigned)
	jls	0x1f	0b00 01 1111	- | reg	-	2 Byte	?	jump if less (signed)