	two complement overflow	flash write enable bit	half carry / flow error	interrupt mask	negative	zero	carry	running state	waiting state		
	Flags										
mnemonic	**V**	**F**	**H**	**I**	**N**	**Z**	**C**	**R**	**W**	**param**	**description**
WAIT										*imm16*	wait *imm16* 256Hz
CALL										*addr16*	push current Program Counter and branch to location
JMP										*addr16*	unconditional branch to location
RET										-	return to caller: pop lp (pointer to locals) and pop pc
NOP										-	no operation; if this instruction is the first at $0000, autostart is active
CLRC							0			-	
SETC							1			-	
END								0		-	
GROW										-	adjust value in stack to one byte wider (push 0)
GROWS										-	adjust signed value in stack to one byte wider: push 0 or push $ff
SHRINK										-	adjust value in stack to one byte smaller (AIS 1)
SHRINKS										-	adjust signed value in stack to one byte smaller: AIS 1 and value |= $80
SJMP											JMP to WORD address retrieved from stack
BRA										*rel8*	unconditional branch to relative location
BRN										*rel8*	branch never
BHI						x	x			*rel8*	branch if higher (C|Z =0)
BLS						x	x			*rel8*	branch if lower or same (C|Z =1)
BCC							x			*rel8*	branch if carry clear(BCC) / branch if higher or same(BHS) (C=0)
BCS							x			*rel8*	branch if carry set(BCS) / branch if lower(BLO) (C=1)
BNE						x				*rel8*	branch if not equal (Z=0)
BEQ						x				*rel8*	branch if equal (Z=1)
BHCC			x							*rel8*	branch if HALF carry clear (H=0) (half carry is also err flag for ECMD & seq)
BHCS			x							*rel8*	branch if HALF carry set (H=1) (half carry is also err flag for ECMD & seq)
BPL					x					*rel8*	branch if plus (N=0)
BMI					x					*rel8*	branch if minus (N=1)
BGE	x				x					*rel8*	branch if greater or equal (N=0) ?V
BLT	x					x				*rel8*	branch if less than Z^V
BGT	x				x	x				*rel8*	branch if greater (N=0 & Z=0) ?V
BLE	x				x	x				*rel8*	branch if less or equal N | (Z^V)
PUSHLA										*var*	push the resolved address of Local variable to the stack
AIS										*imm8*	signed value, positive free space on stack, negative allocate space on stack
SUSPEND								0		*imm8*	suspend execution of the process imm8
RESUME								1		*imm8*	resume the execution of the process imm8
START								1		*imm8,imm16*	start process number imm8 at location imm16 (or label) If the process already running it is aborted and context is reinitialised (stack and flag cleared) and program counter assigned to the new location
WRCMD			↕							*nr,len,ref,ref*	WRITE-READ command (used for complex I/O, init and terminate a SEQ command (wrcmd nr, len: len of param and addr mode of ref) (set ERR on error)
WFLASH		↕								*imm*	set / clear write flash flag: 0 disable write to flash (default), 1 enable
CLR	0				0	1				var	Clear content of variable
NEG	↕				↕	↕	↕				Two's Complement (value=-value)
NOT	0				↕	↕	1				One's Complement
SHR	↕				↕	↕	↕				Shift right
SHL	↕				↕	↕	↕				Shift left
INC	↕		↕		↕	↕	↕				increment
DEC	↕		↕		↕	↕	↕				decrement
TST	0				↕	↕					Test value for negative or zero
ADD	↕		↕		↕	↕	↕				Addtion
SUB	↕		↕		↕	↕	↕				substraction
CMP	↕		↕		↕	↕	↕				compare
MOV	0				↕	↕					assign
AND	0				↕	↕					logical AND
BIT	0				↕	↕					same as AND but does not modify destination
OR	0				↕	↕					logical OR
XOR	0				↕	↕					logical exclusive OR
DIV						↕	↕				Division attention: DIV push the remainder (byte) on the stack even for direct memory mode
MUL			0				0				Multiply
PUSH					↕	↕					push value/variable in stack
POP					↕	↕					pop from stack to variable