Instr. ID	mnemonic	Description	operand A	operand B	operand C	sets flags
0x0	ADD	add the value of A and B together and save the result in A	register A	register B	–	carry
0x1	ADC	add the value of A, B and the carry flag together and save the result in A	register A	register B	–	carry
0x2	ADDI	add a constant to the value of register A and save the result in A	register A	less significant byte of constant	more significant byte of constant	carry
0x3	ADCI	add the value of A, a constant and the carry flah together and save the result in A	register A	less significant byte of constant	more significant byte of constant	carry
0x4	SUB	Subtract B from A and save the result in A	register A	register B	–	carry
0x5	SUC	Subtract B and the carry flag from A and save the result in A	register A	register B	–	carry
0x6	SUBI	Subtract a constant from A and save the result in A	register A	less significant byte of constant	more significant byte of constant	carry
0x7	SUCI	Subtract a constant and the carry flag from A and save the result in A	register A	less significant byte of constant	more significant byte of constant	carry
0x8	AND	Logical AND with two registers, save result in register A	register A	register B	–	–
0x9	ANDI	Logical AND with register A and constant, save result in register A	register A	less significant byte of constant	more significant byte of constant	–
0xA	OR	Logical OR with two registers, save result in register A	register A	register B	–	–
0xB	ORI	Logical OR with register A and constant, save result in register A	register A	less significant byte of constant	more significant byte of constant	–
0xC	XOR	Logical XOR with two registers, save result in register A	register A	register B	–	–
0xD	XORI	Logical XOR with register A and constant, save result in register A	register A	less significant byte of constant	more significant byte of constant	–
0xE	COM	set register A to ist one's complement	register A	–	–	carry
0xF	NEG	set register A to its two's complement	register A	–	–	carry
0x10	SBR	Set bit(s) in register (set to one where bits in constant are one)	register A	less significant byte of constant	more significant byte of constant	–
0x11	CBR	Clear bit(s) in register (set to zero where bits in constant are one)	register A	less significant byte of constant	more significant byte of constant	–
0x12	TST	compare the value of register A with 0, signed	register A	–	–	equality, less/greater than
0x13	CMP	compare the values of two registers with each other, signed	register A	register B	–	equality, less/greater than
0x14	CMPI	compare the value of register A with a constant, signed	register A	less significant byte of constant	more significant byte of constant	equality, less/greater than
0x15	CLR	clear register A (set all bits to zero)	register A	–	–	–
0x16	SER	set register A (set all to one)	register A	–	–	–
0x17	MUL	multiply the value of A with the value of B (both unsigned) and save the result in registers R0 and R1 (little-endian)	register A	register B	–	–
0x18	MULS	multiply the value of A with the value of B (both signed) and save the result in registers R0 and R1 (little-endian)	register A	register B	–	–
0x19	MULSU	multiply the value of A (signed) with the value of B (unsigned) and save the result in registers R0 and R1 (little-endian)	register A	register B	–	–
0x1A	LSL	logical shift left (all bits in register, store result in register A)	register A	–	–	carry
0x1B	LSR	logical shift right (all bits in register, store result in register A)	register A	–	–	carry
0x1C	ROL	rotate left through carry (all bits in register, store result in register A)	register A	–	–	carry
0x1D	ROR	rotate right through carry (all bits in register, store result in register A)	register A	–	–	carry
0x1E	ASR	arithmetic shift right (same as RSR, but preserves the most significant bit)	register A	–	–	carry
0x1F	SWAP	swap the nibbles of the least significant byte in register A, saves the result in register A	register A	–	–	–
0x20	FSET	sets the given flag to 1 (flags: 0/C = carry, 1/E: equality 2/G: greater than, 3/S: bit copy store)	flag name/no.	–	–	as specified in operand A
0x21	FCLR	sets the given flag to 0 (flags: 0/C = carry, 1/E: equality 2/G: greater than, 3/S: bit copy store)	flag name/no.	–	–	as specified in operand A
0x22	BST	save a bit from register A to the bit copy store (flag S)	register A	bit no. (0 to 15)	–	bit copy store
0x23	BLD	load a bit from the bit copy store (flag S) to the specified position in register A	register A	bit no. (0 to 15)	–	–
0x24	MOV	copy the value from register B to register A	register A	register B	–	–
0x25	LDI	load a constant into register A	register A	less significant byte of constant	more significant byte of constant	–
0x26	LDR	load the value of the memory address described by registers B and C (little-endian) into register A	register A	register B (less significant bytes)	register C (more significant bytes)	–
0x27	STR	store the value of register A at the memory address described by the values of register B and C (little-endian)	register A	register B (less significant bytes)	register C (more significant bytes)	–
0x28	PUSH	Push the value of register A to the stack	register A	–	–	–
0x29	POP	Pop a value from the stack to register A	register A	–	–	–
0x2A	SEB	Set baud rate for Serial communication to the value of register A and B (little-endian)	register A	register B	–	–
0x2B	SEBI	Set baud rate for Serial communication to a 3 byte unsigned constant	byte 1	byte 2	byte 3 (most significant byte)	–
0x2C	SOUT	output the least significant byte of register A on Serial	register A	–	–	–
0x2D	SOUTI	output a constant (byte) on Serial	byte for output	–	–	–
0x2E	SIN	read a byte from Serial to register A	register A	–	–	–
0x2F	RJMP	relative jump by signed 24-bit integer constant (little-endian)	byte 1	byte 2	byte 3	–
0x30	JMP	absolute jump to the address described by the unsigned integer combination of the values of registers A and B (32-bit)	register A	register B (less significant bytes)	–	–
0x31	JMPI	absolute jump to the address described by the 24-bit unsigned integer constant (little-endian)	byte 1	byte 2	byte 3 (most significant byte)	–
0x32	CALL	jump to the address described by register A and B (32-bit unsigned integer, little-endian) and push PC to the stack	register A	register B	–	–
0x33	CALLI	jump to the address described by constant (24-it unsigned integer, little-endian) and push PC to the stack	byte 1	byte 2	byte 3 (most significant byte)	–
0x34	RET	Pop a value from the stack to PC (program counter), should be used in combination with CALL and/or CALLI	–	–	–	–
0x35	SEQ	Skip the next instruction if the equal flag is set	–	–	–	–
0x36	SNE	Skip the next instruction if the equal flag is not set	–	–	–	–
0x37	SGR	Skip the next instruction if the greater than flag is set	–	–	–	–
0x38	SLE	Skip the next instruction if the greater than flag is not set	–	–	–	–
0x39	SEQGR	Skip the next instruction if the equal or greater than flag is set	–	–	–	–
0x3A	SEQLE	Skip the next instruction if the equal flag is set or the greater than flag is not set	–	–	–	–
0x3B	BREQ	Jump to the address described by bytes 1-3 (unsigned intger, 24-bit, little-endian) if the equal flag is set	byte 1	byte 2	byte 3 (most significant byte)	–
0x3C	BRNE	Jump to the address described by bytes 1-3 (unsigned intger, 24-bit, little-endian) if the equal flag is not set	byte 1	byte 2	byte 3 (most significant byte)	–
0x3D	BRGR	Jump to the address described by bytes 1-3 (unsigned intger, 24-bit, little-endian) if the greater than flag is set	byte 1	byte 2	byte 3 (most significant byte)	–
0x3E	BRLE	Jump to the address described by bytes 1-3 (unsigned intger, 24-bit, little-endian) if the greater than flag is not set	byte 1	byte 2	byte 3 (most significant byte)	–
0x3F	BREQGR	Jump to the address described by bytes 1-3 (uint, 24-bit, little-endian) if equal flag or greater than flag is set	byte 1	byte 2	byte 3 (most significant byte)	–
0x40	BREQLE	Jump to the address described by bytes 1-3 (uint, 24-bit, little-endian) if equal flag set or greater than flag is not set	byte 1	byte 2	byte 3 (most significant byte)	–
0x41	RBREQ	Jump to the address described by registers A and B (32-bit uint, little-endian) if the equal flag is set	register A	register B	–	–
0x42	RBRNE	Jump to the address described by registers A and B (32-bit uint, little-endian) if the equal flag is not set	register A	register B	–	–
0x43	RBRGR	Jump to the address described by registers A and B (32-bit uint, little-endian) if the greater than flag is set	register A	register B	–	–
0x44	RBRLE	Jump to the address described by registers A and B (32-bit uint, little-endian) if the greater than flag is not set	register A	register B	–	–
0x45	RBREQGR	Jump to the address described by registers A and B (32-bit uint, little-endian) if equal flag or greater than flag is set	register A	register B	–	–
0x46	RBREQLE	Jump to address described by registers A and B (32-bit uint, little-endian) if equal flag set or greater than flag is not set	register A	register B	–	–
0x47	PXL	draw a single pixel at X=register A; Y=register B; color=register C	register A	register B	register C	–
0x48	SCLR	Fill the whole screen with the color of the value of register A	register A	–	–	–
0x49	SCLRI	Fill the whole screen with a constant color (values of bytes 1 and 2, little-endian)	byte 1	byte 2 (most significant byte)	–	–
0x4A	TSIZ	Set text size to the value of register A (least significant byte only)	register A	–	–	–
0x4B	TSIZI	Set text size to a constant	byte 1	–	–	–
0x4C	TCOL	Set text color to the value of register A	register A	–	–	–
0x4D	TCOLB	Set text color to the value of register A and background color to the value of register B	register A	register B	–	–
0x4E	TCOLI	Set text color to vaue of 16-bit unsigned constant	byte 1	byte 2 (most significant byte)	–	–
0x4F	TWRAP	Set text wrap to true if the value of register A is not equal to zero (A != 0)	register A	–	–	–
0x50	TWRAPI	Set text wrap to true if the value of byte 1 is not equal to zero (byte 1 != 0)	byte 1	–	–	–
0x51	TCPOS	Set the cursor position for text to the values of register A (X) and register B (Y) (unsigned integer)	register A	register B	–	–
0x52	TOUT	Print a character (least significant byte of register A) to the screen	register A	–	–	–
0x53	TOUTI	Print a character (byte 1)	byte 1	–	–	–
0x54	IMG	Draw an image file with it's path specified by a null-terminated string at a memory address specified by registers A & B at the cursor position	register A	register B (most significant bytes)	–	–
0x55	IMGI	Draw an image file with it's path specified by a null-terminated string at a memory address specified by bytes 1-3 at the current cursor position	byte 1	byte 2	byte 3 (most significant byte)	–
0x56	FEXISTS	Set register A to 1, if file with path located at memory address specified by reg B & C exists (null-terminated string)	register A	register B	register C	–
0x57	FMKDIR	Set reg A to 1, if directory with file path located at memory address specified reg by B & C exists (null-terminated str)	register A	register B	register C	–
0x58	FOPEN	Open file with path located at memory address specified reg by B & C exists (null-terminated str), if unsuccessful set A to 0	register A	register B	register C	–
0x59	FREM	Set reg A to 1, if the file with path located at memory address specified reg by B & C could be deleted (null-terminated str)	register A	register B	register C	–
0x5A	FRMDIR	Set reg A to 1, if the directory with path located at memory address specified reg by B & C could be deleted (null-terminated str)	register A	register B	register C	–
0x5B	FNAME	Set register A to the byte value of the Xth char in the filename (X = value of register B)	register A	register B	–	–
0x5C	FAV	Set register A to the number of bytes available to read	register A	–	–	–
0x5D	FCLOS	close the currently opened file, automatically invoked by open	–	–	–	–
0x5E	FFLUS	flush the file (ensure that all written bytes are also physically written to SD card)	–	–	–	–
0x5F	FPEK	read a single byte from the SD card to register A without advancing to the next one	register A	–	–	–
0x60	FPOS	save the current position in the file to registers 0 and 1 (little endian, unsigned long)	–	–	–	–
0x61	FSEK	go to a position in the file specified by registers B and C (little-endian, unsigned int), set reg A to 1 on success, to 0 on failure	register A	register B	register C (most significant bytes)	–
0x62	FSEKI	go to a position in the file specified by byte 1-3 (sets bit copy store to 1 on success, to 0 on failure)	byte 1	byte 2	byte 3 (most significant byte)	bit copy store
0x63	FISDIR	set register a to 1, if the currently open file is a directory, otherwise set register A to 0	register A	–	–	–
0x64	FNEXT	opens the next file in the current directory (sets bit copy store to 1 on success, to 0 on failure); will always switch the current file	–	–	–	–
0x65	FREW	return to the first file in the directory (opens the first file on next call to FNEXT)	–	–	–	–
0x66	FOUT	write the least significant byte of register A to the file	register A	–	–	–
0x67	FOUTI	write a constant byte to the file	byte 1	–	–	–
0x68	FIN	read a byte from the file to register A	register A	–	–	–
0x69	SET	set a system variable (least significant byte of reg A) to the values of registers B and C (register C might not be accessed depending on the variable)	register A	register B (least significant bytes)	register C (most significant bytes)	–
0x6A	SETI	set a system variable to the values of registers A and B (register B might not be accessed depending on the variable)	sysvar ID	register A (least significant bytes)	register B (most significant bytes)	
0x6B	GET	get a system variable, save to registers 0 and 1	register A	–	–	–
0x6C	GETI	get a system variable, save to registers 0 and 1	sysvar ID	–	–	–
0x6D	RUN	change the currently running program to the one with it's executable path described by the null-terminated string at memory address by reg A & B	register A	register B (most significant byte)		
0x6E						
0x6F						
0x70						
0x71						
0x72						
0x73						
0x74						
0x75						
0x76						
0x77						
0x78						
0x79						
0x7A						
0x7B						
0x7C						
0x7D						
0x7E						
0x7F						
0x80						
0x81						
0x82						
0x83						
0x84						
0x85						
0x86						
0x87						
0x88						
0x89						
0x8A						
0x8B						
0x8C						

0x8D							
0x8E							
0x8F							
0x90							
0x91							
0x92							
0x93							
0x94							
0x95							
0x96							
0x97							
0x98							
0x99							
0x9A							
0x9B							
0x9C							
0x9D							
0x9E							
0x9F							
0xA0							
0xA1							
0xA2							
0xA3							
0xA4							
0xA5							
0xA6							
0xA7							
0xA8							
0xA9							
0xAA							
0xAB							
0xAC							
0xAD							
0xAE							
0xAF							
0xB0							
0xB1							
0xB2							
0xB3							
0xB4							
0xB5							
0xB6							
0xB7							
0xB8							
0xB9							
0xBA							
0xBB							
0xBC							
0xBD							
0xBE							
0xBF							
0xC0							
0xC1							
0xC2							
0xC3							
0xC4							
0xC5							
0xC6							
0xC7							
0xC8							
0xC9							
0xCA							
0xCB							
0xCC							
0xCD							
0xCE							
0xCF							
0xD0							
0xD1							
0xD2							
0xD3							
0xD4							
0xD5							
0xD6							
0xD7							
0xD8							
0xD9							
0xDA							
0xDB							
0xDC							
0xDD							
0xDE							
0xDF							
0xE0							
0xE1							
0xE2							
0xE3							
0xE4							
0xE5							
0xE6							
0xE7							
0xE8							
0xE9							
0xEA							
0xEB							
0xEC							
0xED							
0xEE							
0xEF							
0xF0							
0xF1							
0xF2							
0xF3							
0xF4							
0xF5							
0xF6							
0xF7							
0xF8							
0xF9							
0xFA							
0xFB							
0xFC							
0xFD							
0xFE							
0xFF							