Table 1

						Word 1																	Word 2																
						HEX	2nd	OPCODE									register Y				register X				Immediate Data (kk/pp/aaa)														
	Instruction	Description	Function	ZERO	CARRY		15	14	13	12	11	10	9	8		7	6	5	4	3	2	1	0	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	**NOP**	Do Nothing		-	-	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	none															
1	**ADD rX, kk**	Add register rX with literal kk	rX <= rX + kk	?	?	2	1	0	0	0	0	0	1	0	2	0	0	0	0	x	x	x	x	kk															
2	**ADD rX, rY**	Add register rX with register rY	rX <= rX + rY	?	?	4	0	0	0	0	0	1	0	0	4	y	y	y	y	x	x	x	x	none															
3	**ADDCY rx, kk**	Add register rX with literal kk with CARRY	rX <= rX + kk + CARRY	?	?	6	1	0	0	0	0	1	1	0	6	0	0	0	0	x	x	x	x	kk															
4	**ADDCY rX, rY**	Add register rX with register rY with CARRY	rX <= rX + rY + CARRY	?	?	8	0	0	0	0	1	0	0	0	8	y	y	y	y	x	x	x	x	none															
5	**AND rX, kk**	Bitwise AND register rX with literal kk	rX <= rX & kk	?	?	A	1	0	0	0	1	0	1	0	10	0	0	0	0	x	x	x	x	kk															
6	**AND rX, rY**	Bitwise AND register rX with register rY	rX <= rX & rY	?	?	C	0	0	0	0	1	1	0	0	12	y	y	y	y	x	x	x	x	none															
7	**CALL aaa**	Unconditionally call subroutine at aaa	TOS <= PC; PC <= aaa	-	-	E	1	0	0	0	1	1	1	0	14	0	0	0	0	0	0	0	0	aaa															
8	**CALLC aaa**	If CARRY call subroutine at aaa	if CARRY TOS <= PC; PC <= aaa	-	-	10	1	0	0	1	0	0	0	0	16	0	0	0	0	0	0	0	0	aaa															
9	**CALLNC aaa**	If NOT CARRY call subroutine at aaa	if !CARRY TOS <= PC; PC <= aaa	-	-	12	1	0	0	1	0	0	1	0	18	0	0	0	0	0	0	0	0	aaa															
10	**CALLZ aaa**	If ZERO call subroutine at aaa	if ZERO TOS <= PC; PC <= aaa	-	-	14	1	0	0	1	0	1	0	0	20	0	0	0	0	0	0	0	0	aaa															
11	**CALLNZ aaa**	If NOT ZERO call subroutine at aaa	if !ZERO TOS <= PC; PC <= aaa	-	-	16	1	0	0	1	0	1	1	0	22	0	0	0	0	0	0	0	0	aaa															
12	**COMP rX, kk**	Compare register rX with literal kk.	If rX== kk then ZERO =1; if rx <kk then CARRY = 1	?	?	18	1	0	0	1	1	0	0	0	24	0	0	0	0	x	x	x	x	kk															
13	**COMP rX, rY**	Compare register rX with register rY	If rX== rY then ZERO =1; if rx <rY then CARRY = 1	?	?	1A	0	0	0	1	1	0	1	0	26	y	y	y	y	x	x	x	x	none															
14	**DISINT**	Disable interrupt input	Interrupt Enable = 0	-	-	1C	0	0	0	1	1	1	0	0	28	0	0	0	0	0	0	0	0	none															
15	**ENINT**	Enable interrupt input	Interrupt Enable = 1	-	-	1E	0	0	0	1	1	1	1	0	30	0	0	0	0	0	0	0	0	none															
16	**INPUT rX, (rY)**	Read input port pointed to by rY into register rX	rX <= PORT(rY)	-	-	20	0	0	1	0	0	0	0	0	32	y	y	y	y	x	x	x	x	none															
17	**INPUT rx, pp**	Read input port pp into register rX	rX <= PORT(pp)	-	-	22	1	0	1	0	0	0	1	0	34	0	0	0	0	x	x	x	x	pp															
18	**JUMP aaa**	Unconditionally jump to aaa	PC <= aaa	-	-	24	1	0	1	0	0	1	0	0	36	0	0	0	0	0	0	0	0	aaa															
19	**JUMPC aaa**	Jump if CARRY to aaa	if CARRY PC <= aaa	-	-	26	1	0	1	0	0	1	1	0	38	0	0	0	0	0	0	0	0	aaa															
20	**JUMPNC aaa**	Jump if NOT CARRY to aaa	if !CARRY PC <= aaa	-	-	28	1	0	1	0	1	0	0	0	40	0	0	0	0	0	0	0	0	aaa															
21	**JUMPZ aaa**	Jump if ZERO to aaa	if ZERO PC <= aaa	-	-	2A	1	0	1	0	1	0	1	0	42	0	0	0	0	0	0	0	0	aaa															
22	**JUMPNZ aaa**	Jump if NOT ZERO to aaa	if ! ZERO PC <= aaa	-	-	2C	1	0	1	0	1	1	0	0	44	0	0	0	0	0	0	0	0	aaa															
23	**LOAD rX, kk**	Load register rX with literal kk	rX <= kk	-	-	2E	1	0	1	0	1	1	1	0	46	0	0	0	0	x	x	x	x	kk															
24	**LOAD rX, rY**	Load register rX with contents of register rY	rX <= rY	-	-	30	0	0	1	1	0	0	0	0	48	y	y	y	y	x	x	x	x	none															
25	**OR rX, kk**	Logically OR register rX with literal kk	rX <= rX | kk	?	0	32	1	0	1	1	0	0	1	0	50	0	0	0	0	x	x	x	x	kk															
26	**OR rX, rY**	Logically OR register rX with register rY	rX <= rX | rY	?	0	34	0	0	1	1	0	1	0	0	52	y	y	y	y	x	x	x	x	none															
27	**OUTPUT rX, (rY)**	Write register rX to port pointed to by register rY	PORT(rY) <= rX	-	-	36	0	0	1	1	0	1	1	0	54	0	0	0	0	0	0	0	0	none															
28	**OUTPUT rX, pp**	Write register rX to port pp	PORT(pp) <= rX	-	-	38	1	0	1	1	1	0	0	0	56	0	0	0	0	x	x	x	x	pp															
29	**RETURN**	Unconditionally return from subroutine	PC <= TOS+1	-	-	3A	0	0	1	1	1	0	1	0	58	0	0	0	0	0	0	0	0	none															
30	**RETURNC**	If CARRY return from subroutine	if CARRY PC <= TOS+1	-	-	3C	0	0	1	1	1	1	0	0	60	0	0	0	0	0	0	0	0	none															
31	**RETURNNC**	If NOT CARRY return from subroutine	if !CARRY PC <= TOS+1	-	-	3E	0	0	1	1	1	1	1	0	62	0	0	0	0	0	0	0	0	none															
32	**RETURNZ**	if ZERO return from subroutine	if ZERO PC <= TOS+1	-	-	40	0	1	0	0	0	0	0	0	64	0	0	0	0	0	0	0	0	none															
33	**RETURNNZ**	if NOT ZERO return from subroutine	if !ZERO PC <= TOS+1	-	-	42	0	1	0	0	0	0	1	0	66	0	0	0	0	0	0	0	0	none															
34	**RETDIS**	Return from interrupt with interrupts disabled	PC <= TOS+1; Interrupt Enable <= 0	?	?	44	0	1	0	0	0	1	0	0	68	0	0	0	0	0	0	0	0	none															
35	**RETEN**	Return from interrupt with interrupts enabled	PC <= TOS+1; Interrupt Enable <= 1	?	?	46	0	1	0	0	0	1	1	0	70	0	0	0	0	0	0	0	0	none															
36	**RL rX**	Rotate register rX left	rX <= {rX[14:0],rX[15]}; CARRY <= rX[15]	?	?	48	0	1	0	0	1	0	0	0	72	0	0	0	0	x	x	x	x	none															
37	**RR rX**	Rotate register rX right	rX <= {rX[0],rX[15:1]}; CARRY <= rX[0]	?	?	4A	0	1	0	0	1	0	1	0	74	0	0	0	0	x	x	x	x	none															
38	**SL0 rX**	Shift register rX left, zero fill	rX <= {rX[14:0],0}; CARRY <= rX[15]	?	?	4C	0	1	0	0	1	1	0	0	76	0	0	0	0	x	x	x	x	none															
39	**SL1 rX**	Shift register rX left, one fill	rX <= {rX[14:0],1}; CARRY <= rX[15]	0	?	4E	0	1	0	0	1	1	1	0	78	0	0	0	0	x	x	x	x	none															
40	**SLA rX**	Shift register rX left through all bits, include CARRY	rX <= {rX[14:0], CARRY}; CARRY <= rX[15]	?	?	50	0	1	0	1	0	0	0	0	80	0	0	0	0	x	x	x	x	none															
41	**SLX rX**	Shift register rX left. Bit rX[0] is unaffected	rX <= {rX[14:0],rX[0]}; CARRY <= rX[15]	?	?	52	0	1	0	1	0	0	1	0	82	0	0	0	0	x	x	x	x	none															
42	**SR0 rX**	Shift register rX right, zero fill	rX <= {0,rX[15:1]}; CARRY <= rX[0]	?	?	54	0	1	0	1	0	1	0	0	84	0	0	0	0	x	x	x	x	none															
43	**SR1 rX**	Shift register rX right, one fill	rX <= {1,rX[15:1]}; CARRY <= rX[0]	?	?	56	0	1	0	1	0	1	1	0	86	0	0	0	0	x	x	x	x	none															
44	**SRA rX**	Shift register rX right through all bits, include CARRY	rX <= {CARRY,rX[15:1]}; CARRY <= rX[0]	?	?	58	0	1	0	1	1	0	0	0	88	0	0	0	0	x	x	x	x	none															
45	**SRX rX**	Shift register rX right. Bit rX[7] is unaffected	rX <= {rX[15],rX[15:1]}; CARRY <= rX[0]	?	?	5A	0	1	0	1	1	0	1	0	90	0	0	0	0	x	x	x	x	none															
46	**SUB rX, kk**	Subtract literal kk from register rX	rX <= rX - kk	?	?	5C	1	1	0	1	1	1	0	0	92	0	0	0	0	x	x	x	x	kk															
47	**SUB rX, rY**	Subtract register rY from register rX	rX <= rX - rY	?	?	5E	0	1	0	1	1	1	1	0	94	y	y	y	y	x	x	x	x	none															
48	**SUBC rX, kk**	Subtract literal kk from register rX with CARRY	rX <= rX - kk - CARRY	?	?	60	1	1	1	0	0	0	0	0	96	0	0	0	0	x	x	x	x	kk															
49	**SUBC rX, rY**	Subtract register rY from register rX with CARRY	rX <= rX - rY - CARRY	?	?	62	0	1	1	0	0	0	1	0	98	y	y	y	y	x	x	x	x	none															
50	**TEST rX, kk**	Test bits in register rX against literal kk	If (rX & kk) == 0; ZERO <= 1; CARRY <= ODD PARITY (rX & kk)	?	?	64	1	1	1	0	0	1	0	0	100	0	0	0	0	x	x	x	x	kk															
51	**TEST rX, rY**	Test bits in register rX against register rY	If (rX & rY) == 0; ZERO <= 1; CARRY <= ODD PARITY (rX & rY)	?	?	66	0	1	1	0	0	1	1	0	102	y	y	y	y	x	x	x	x	none															
52	**XOR rX, kk**	Bitwise XOR register rX with literal kk	rX <= rX ^ kk	?	0	68	1	1	1	0	1	0	0	0	104	0	0	0	0	x	x	x	x	kk															
53	**XOR rX, rY**	Bitwise XOR register rX with register rY	rX <= rX ^ rY	?	0	6A	0	1	1	0	1	0	1	0	106	y	y	y	y	x	x	x	x	none															
54	**FETCH rX, kk**	Read scratchpad RAM location kk into register rX	rX <- RAM[kk]	-	-	70	1	1	1	1	0	0	0	0	112	0	0	0	0	x	x	x	x	kk															
55	**FETCH rX, (rY)**	Read scratchpad RAM pointed to by rY into register rX	rX <- RAM[(rY)]	-	-	72	0	1	1	1	0	0	1	0	114	y	y	y	y	x	x	x	x	none															
56	**STORE rX, kk**	Write register rX to scratchpad RAM location kk	RAM[kk] <- rX	-	-	74	1	1	1	1	0	1	0	0	116	0	0	0	0	x	x	x	x	kk															
57	**STORE rX, (rY)**	Write register rX to scratchpad RAM pointed to by rY	RAM[(rY)]<- rX	-	-	76	0	1	1	1	0	1	1	0	118	y	y	y	y	x	x	x	x	none															
				? in Carry/Zero field means the bit will be set/reset depending on the result of the operation **- in Carry/Zero field means the bit will not be modified as a result of the operation** **0 in Carry/Zero field means the bit will reset as a result of the operation**																																				
		Updated 11/7/2017 - SUBCY to SUBC																																					