OPCODE	Action	Note	IMD. ARG1	IMD. ARG2	IMD. BOTH
0	ADD	ARG1 + ARG2	128	64	192
1	SUB	ARG1 - ARG2	129	65	193
2	AND	ARG1 AND ARG2	130	66	194
3	OR	ARG1 OR ARG2	131	67	195
4	NOT A	NOT A, ignores ARG2	132	68	196
5	XOR	ARG1 XOR ARG2	133	69	197
6	MULTIPLY	ARG1 * ARG2	134	70	198
7	DIV	ARG1 / ARG2	135	71	199
8	MOD	ARG1 % ARG2	136	72	200
9	SHL	ARG1 Shift left by ARG2	137	73	201
10	SHR	ARG1 Shift right by ARG2	138	74	202
11	ASHR	ARG1 Shift right by ARG2 (keep signed)	139	75	203
12	ROL	ARG1 Rotate left by ARG2	140	76	204
13	ROR	ARG1 Rotate right by ARG2	141	77	205
14			142	78	206
15			143	79	207
16	IF_EQL	IF ARG1 == ARG2	144	80	208
17	IF_NEQ	IF ARG1 != ARG2	145	81	209
18	IF_LES	IF ARG1 < ARG2	146	82	210
19	IF_LOE	IF ARG1 <= ARG2	147	83	211
20	IF_GRT	IF ARG1 > ARG2	148	84	212
21	IF_GOE	IF ARG1 >= ARG2	149	85	213
57			185	121	249

Keyword	Type	Decimal	Note	Usage example
ADD	OPCODE	0	ARG1 + ARG2	
ADDRESS	SRC(ARG1)	5	For readability (use with other keywords)	Refer to keyword "NEXT/PREVIOUS"
AND	OPCODE	2	ARG1 && ARG2 (Bitwise)	
ASHR	OPCODE	11	ARG1 Shift right by ARG2 (keep signed)	
BACK	SRC(ARG1)	8	For readability (use with other keywords)	Refer to keyword "RETURN"
CALL	OPCODE	210	Call a subroutine (same as JMP)	CALL NOW SUBROUTINE functionA
CALLER	DES	6	For readability (use with other keywords)	Refer to keyword "RETURN"
COPY_CONST	OPCODE	192	Copy a literal to destination	COPY_CONST 125 TO OUTPUT
COUNTER	DES	6	Program counter	
DIV	OPCODE	7	ARG1 / ARG2	
HALT	OPCODE	63	Stop the program	HALT
IF_EQL	OPCODE	16	ARG1 = ARG2	IF_EQL REG0 REG1 lab_loop
IF_GOE	OPCODE	21	ARG1 >= ARG2	IF_GOE REG0 REG1 lab_loop
IF_GRT	OPCODE	20	ARG1 > ARG2	IF_GRT REG0 REG1 lab_loop
IF_LES	OPCODE	18	ARG1 < ARG2	IF_LES REG0 REG1 lab_loop
IF_LOE	OPCODE	19	ARG1 <= ARG2	IF_LOE REG0 REG1 lab_loop
IF_NEQ	OPCODE	17	ARG1 != ARG2	IF_NEQ REG0 REG1 lab_loop
IMD_ARG1	OPCODE	128	Treat ARG1 as literal, use with OR operator (|)	ADD|IMDARG1 150 REG0 REG1
IMD_ARG2	OPCODE	64	Treat ARG2 as literal, use with OR operator (|)	IF_LES|IMDARG2 REG0 150 lab_loop
IMD_ARG3	OPCODE	192	Treat ARG1 and ARG2 as literal, use with OR operator (|)	SUB|IMDARG2 150 50 REG1
IN	SRC(ARG2)	1	For readability (use with other keywords)	Refer to keyword "NEXT/PREVIOUS"
INPUT	SRC(ARG1)	7	In-game only	
JMP	OPCODE	208	For readability, work as a GOTO	JMP NOW TO lab_loop
MEMORY	DES	5	For readability (use with other keywords)	Refer to keyword "NEXT/PREVIOUS"
MOD	OPCODE	8	ARG1 % ARG2	
MOV	OPCODE	64	Copy the value of a "variable" to a destination	MOV REG0 TO OUTPUT
MULTIPLY	OPCODE	6	ARG1 * ARG2	
NEXT	OPCODE	64	Increment address/value	NEXT ADDRESS IN MEMORY; NEXT REG0 IN REG0
NOT_A	OPCODE	4	NOT A, ignores ARG2 (Bitwise)	
NOW	SRC(ARG1)	0	For readability (use with other keywords)	
OR	OPCODE	3	ARG1 || ARG2 (Bitwise)	
OUTPUT	DES	7	In-game only	
PREVIOUS	OPCODE	65	Decrement address/value	PREVIOUS ADDRESS IN MEMORY; PREVIOUS REG0 IN REG0
RAM	SRC/DES	128	Get/send values from/to RAM	MOV REG0 TO RAM; MOV RAM TO REG0
REG0	SRC/DES	0	First register, work as a "variable"	
REG1	SRC/DES	1	Second register, work as a "variable"	
REG2	SRC/DES	2	Third register, work as a "variable"	
REG3	SRC/DES	3	Fourth register, work as a "variable"	
REG4	SRC/DES	4	Fifth register, work as a "variable"	
REG5	SRC/DES	5	Sixth register, work as a "variable"	
RETURN	OPCODE	64	Return from a subroutine	RETURN BACK TO CALLER
ROL	OPCODE	12	Rotate bits of ARG1 to left by ARG2	
ROR	OPCODE	13	Rotate bits of ARG1 to right by ARG2	
SHL	OPCODE	9	Bit shift left ARG1 by ARG2	
SHR	OPCODE	10	Bit shift right ARG1 by ARG2	
STACK	DES	8	Stack data structure	
SUB	OPCODE	1	ARG1 - ARG2	
SUBROUTINE	SRC(ARG2)	255	For readability (use with other keywords)	Refer to keyword "CALL"
TO	SRC(ARG2)	0	For readability (use with other keywords)	Refer to keyword "RETURN"
XOR	OPCODE	5	ARG1 ^ ARG2	