Instruction set

Category	Opcode	Hexadecimal	Instruction	Arguments		Operation
JUMP	0000	0	jump x	flag (3)	value (1)	see flags
STACK	0001	1	/	mode (2)	reg (2)	(EF) push/pop reg
MEMORY	0010	2	mget	r1 (2)	r2 (2)	r2 ← @ r1
	0011	3	mset	r1 (2)	r2 (2)	@ r1 ← r2
INPUT / OUTPUT	0100	4	iget	bus (4)		A ← @ bus
	0101	5	oset	bus (4)		@ bus ← A
INSTANT VALUE	0110	6	hiV	value (4)		V[4 - 7] ← value
	0111	7	loV	value (4)		V[0 - 3] ← value
ARITHMETIC	1000	8	add	r1 (2)	r2 (2)	(COPNZ) X ← r1 + r2
	1001	9	sub	r1 (2)	r2 (2)	(COPNZ) X ← r2 - r1
LOGIC	1010	a	/	mode (2)	r2 (2)	(COPNZ) X ← shift/rot r2
	1011	b	nand	r1 (2)	r2 (2)	(COPNZ) X ← ¬ (r1 ∧ r2)
	1100	c	or	r1 (2)	r2 (2)	(COPNZ) X ← r1 ∨ r2
	1101	d	xor	r1 (2)	r2 (2)	(COPNZ) X ← r1 ⊕ r2
COMPARISON	1110	e	les	r1 (2)	r2 (2)	(COPNZ) X ← (r2 - r1 < 0)
	1111	f	leq	r1 (2)	r2 (2)	(COPNZ) X ← (r2 - r1 ≤ 0)

UAL registers (2)	
00	A
01	B
10	V
11	X

input registers (2)	
00	SP
01	MPA
10	CPC
11	CPA

output registers (2)	
00	SP
01	MPA
10	JPC
11	JPA

flags (3)		
000	_	nop (0), goto (1)
001	C	carry flag
010	O	overflow flag
011	P	parity flag
100	N	negative result flag
101	Z	result zero flag
110	E	empty stack flag
111	F	full stack flag
if flag == value : CPC ← JPC, CPA ← JPA		

shift/rot mode (2)	
00	shl
01	shr
10	rol
11	ror

push/pop mode (2)		
00	psh UAL reg	(EF) SP ← SP + 1
01	psh input reg	stack ← register
10	pop UAL reg	(EF) SP ← SP - 1
11	pop output reg	register ← stack