Single Instruction Format Processor (update 1)
https://hackaday.io/project/173996-sifp-single-instruction-format-processor

All instructions are 16-bit, and follow the same format below

Instruction field:	15..12	11..9	8..6	5..3	2..0	
Target register:	P Program counter	A Accumulator	X Index register X	Y Index register Y	S Stack pointer	Octal values For A, X ,Y , S
0 (default)	**NOP**	**NOA** --	**NOX** --	**NOY** --	**NOS** --	0 (default)
1	**M[IMM]** (p++)	**SBC** cz	**CPX** cz	**CPY** cz	**CPS** cz	1
2	**BRANCH** (p += m[p])	**XOR** -z	**INX** cz	**INY** cz	**M[POP]** Cz (M[S+])	2
3	**JUMP** (p = m[p])	**RRC** cz	**DEX** cz	**DEY** cz	**M[PUSH]** Cz (M[-S])	3
4	**LDP** (p = data)	**LDA** -z	**LDX** -z	**LDY** -z	**LDS** -z	4
5	**STP4** (data = p + 4) P4	**ADC** cz	**ADX** cz	**ADY** cz	**ADS** cz	5
6	**STP2** (data = p + 2)	**AND** -z	**M[X]** --	**M[Y]** --	**M[S]** --	6
7	**STP** (data = p)	**STA** --	**STX** --	**STY** --	**STS** --	7
8	**BAC** (p += (ac ? m[p] : 1))					
9	**BAZ** (p += (az ? m[p] : 1))					
A	**BXC** (p += (xc ? m[p] : 1))					
B	**BXZ** (p += (xz ? m[p] : 1))					
C	**BYC** (p += (yc ? m[p] : 1))					
D	**BYZ** (p += (yz ? m[p] : 1))					
E	**BSC** (p += (sc ? m[p] : 1))					
F	**BSZ** (p += (sz ? m[p] : 1))					

Notes:

- Registers A, X, Y, S have own independent Carry and Zero flags, which can be tested using B?C and B?Z branch instructions
- 8 flags are stored in F register, which can only be stored as stack push or loaded as stack pop
- Any of these operations generates VMA (valid memory address). If more than one are in same instruction, values are ADDed.
- Any of these operations generates RnW low (write to memory), if VMA is also true (STPx allows storing program counter with small offset). If more than one are in same instructions, values are OR'd.
- Any of these operations loads from internal data bus (which also has external memory bus as one input)
- Internal operations, no data/address bus interaction
- Each instruction is a vector of 5 values (one per register), for example: "STA, INX, M[PUSH];" pushes A to stack while incrementing X
- F(lags) register (note register flags matching B?C and B?Z op-codes):

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
IE	TE							AC	XC	YC	SC	AZ	XZ	YZ	SZ
Enable interrupts when 1	Enable trace when 1	Reserved for future use						Carry flags per register				Zero flags per register			