Single Instruction Format Processor

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	**NOP**	**NOA** --	**NOX** --	**NOY** --	**NOS** --	0
1	**LDP**	**LDA** -z	**CPX** cz	**CPY** cz	**CPS** cz	1
2	**ADP**	**XOR** -z	**INX** cz	**INY** cz	**M[S++]** cz	2
3	**P2**	**SLC** cz	**DEX** cz	**DEY** cz	**M[--S]** cz	3
4	**P3**	**SRC** cz	**LDX** -z	**LDY** -z	**LDS** -z	4
5	**P4**	**ADC** cz	**ADX** cz	**ADY** cz	**ADS** cz	5
6	**P0**	**AND** -z	**M[X]** --	**M[Y]** --	**M[S]** --	6
7	**M[P++]**	**STA** --	**STX** --	**STY** --	**STS** --	7
8	**BAC**	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) • Any of these operations generates RnW low (write to memory), if VMA is also true (Px allows storing program counter with small offset) • Any of these operations loads from internal data bus (which is also has external memory bus as one input) • Internal operations, no data/address bus interaction				
9	**BAZ**					
A	**BXC**					
B	**BXZ**					
C	**BYC**					
D	**BYZ**					
E	**BSC**					
F	**BSZ**					