Specifications and Instruction Set

Specifications	
Addressing capacity (bits)	**16**
RAM (kilobits)	**512**
RAM (kilobytes)	**64**
Stack (bytes)	**256**
Interrupt Vectors Table (bytes)	**512**
Instruction size (bits)	**32**
Instruction size (bytes)	**4**
RAM capacity (instructions)	**16192**
Number of instructions	**48**

Flag Register (8b)	
Carry (C)	0
Equal (E)	1
Interrupt (I)*	2
Negative (N)	3
Superior (S)	4
Zero (Z)	5
Inferior (F)	6
Halt (H)	7

** Flag up when interrupts are handable*

Instruction format (4 bytes)			
OpCode (6b)	**A.M. (4b)**	**Regs (6b)**	**V1 (8b) V2 (8b)**

RAM format	Address
Stack (256 bytes)	0x0000 --- 0x00FF
Interrupt Vectors Table (512 bytes)	0x0100 --- 0x02FF
Program Memory (63,25 kB)	0x0300 --- --- --- --- --- --- --- --- --- 0xFFFF

8b Registers	Code
PC	*Prog. Counter*
STK	*Stack Pointer*
A	0x0
B	0x1
C	0x2
D	0x3
I	0x4
J	0x5
X	0x6
Y	0x7

Addressing modes (4b)	
N/A	0
Reg	1
Reg/Imm8	2
Reg/Ram	3
RamReg/ImmReg	4
Reg16	5
Imm24	6
Imm8	7

Instructions	A.M.	DEC	HEX	BIN	Affected flags
NOP	0	0	0	0	
ADC – R1 + R2 → R1	1				
ADC – R1 + V1 → R1	2	1	1	1	
ADC – R1 + $(Vx) → R1	3				C, Z, N
ADD – R1 + R2 → R1	1				
ADD – R1 + V1 → R1	2	2	2	10	
ADD – R1 + $(Vx) → R1	3				
AND – R1 & R2 → R1	1				
AND – R1 & V1 → R1	2	3	3	11	Z, N
AND – R1 & $(Vx) → R1	3				
CAL – $(Rx) → PC	5	4	4	100	C
CAL – $(Vx) → PC	6				C
CLC		5	5	101	C
CLE		6	6	110	E
CLI		7	7	111	I
CLN	0	8	8	1000	N
CLS		9	9	1001	S
CLZ		10	A	1010	Z
CLF		11	B	1011	F
CMP – R1 ? R2	1				
CMP – R1 ? V1	2	12	C	1100	S, E, I, Z
CMP – R3 ? $(Rx)	4				
DEC – R1-- → R1	1				
DEC – $(Rx)-- → $(Rx)	5	13	D	1101	C, Z, N
DEC – $(Vx)-- → $(Vx)	6				
HLT	0	14	E	1110	H, I
IN – Port(R2) → R1		15	F	1111	
OUT – R2 → Port(R1)	1	16	10	10000	
INC – R1++ → R1					
INC – $(Rx)++ → $(Rx)	5	17	11	10001	C, Z, N
INC – $(Vx)++ → $(Vx)	6				
INT – V1	7	18	12	10010	I
IRT	0	19	13	10011	

Instructions	A.M.	DEC	HEX	BIN	Affected flags
JMC – (Rx) → PC	5	20	14	10100	
JMC – (Vx) → PC	6				
JME – (Rx) → PC	5	21	15	10101	
JME – (Vx) → PC	6				
JMN – (Rx) → PC	5	22	16	10110	
JMN – (Vx) → PC	6				
JMP – (Rx) → PC	5	23	17	10111	
JMP – (Vx) → PC	6				
JMS – (Rx) → PC	5	24	18	11000	
JMS – (Vx) → PC	6				
JMZ – (Rx) → PC	5	25	19	11001	
JMZ – (Vx) → PC	6				
JMF – (Rx) → PC	5	26	1A	11010	
JMF – (Vx) → PC	6				
STR – R3 → $(Rx)	4	27	1B	11011	
STR – R1 → $(Vx)	3				
LOD – $(Rx) → R3	4	28	1C	11100	
LOD – $(Vx) → R1	3				
MOV – R2 → R1	1	29	1D	11101	
MOV – V1 → R1	2				
NOT – ~R1 → R1	1	30	1E	11110	Z, N
NOT – ~$(Vx) → $(Vx)	6				
OR – R1 or R2 → R1	1	31	1F	11111	
OR – R1 or V1 → R1	2				
OR – R1 or $(Vx) → R1	3				
POP – $(STK - 1) → R1	1	32	20	100000	C
PSH – R1 → $(STK)		33	21	100001	C
RET	0	34	22	100010	
SHL – R1	1	35	23	100011	C, Z, N
ASR – R1		36	24	100100	
SHR – R1		37	25	100101	
Instructions	**A.M.**	**DEC**	**HEX**	**BIN**	**Affected flags**
STC		38	26	100110	C

STE	0	39	27	100111	E
STI		40	28	101000	I
STN		41	29	101001	N
STS		42	2A	101010	S
STZ		43	2B	101011	Z
STF		44	2C	101100	F
SUB — R1 - R2 → R1	1	45	2D	101101	C, Z, N
SUB — R1 - V1 → R1	2				
SUB — R1 - $(Vx) → R1	3				
SBB — R1 - R2 - 1 → R1	1	46	2E	101110	C, Z, N
SBB — R1 - V1 - 1 → R1	2				
SBB — R1 - $(Vx) — 1 → R1	3				
XOR — R1 ⊕ R2 → R1	1	47	2F	101111	Z, N
XOR — R1 ⊕ V1 → R1	2				
XOR — R1 ⊕ $(Vx) → R1	3				

Completion
0,00%

Implemented

Logisim	Emulator
	X
	X
	X
	X
	X
	X
	X
	X
	X
	X
	X
	X
	X
	X
	X
	X
	X
	X
	X
	X
	X
	X
	X
	X
	X
	X
	X
	X
	X
	X
	X

Implemented	
	X
	X
	X
	X
	X
	X
	X
	X
	X
	X
	X
	X
	X
	X
	X
	X
	X
	X
	X
	X
	X
	X
	X
	X
	X
	X
	X
	X
Implemented	
	X

X
X
X
X
X
X
X
X
X
X
X
X
X
X
X
X