VonSim Instruction Set

Data Transfer Instructions							
Instruction	**Description**	**Flags**					**Obs.**
		CF	ZF	SF	IF	OF	
MOV dest, source	Copies source to dest	—	—	—	—	—	1
PUSH source	Pushes source onto the stack	—	—	—	—	—	2
POP dest	Pops the top of the stack and loads it into dest	—	—	—	—	—	2
PUSHF	Pushes FLAGS	X	X	X	X	X	
POPF	Pops FLAGS	X	X	X	X	X	
IN dest, source	Loads the value from port source into dest	—	—	—	—	—	3
OUT dest, source	Loads the value from source into port dest	—	—	—	—	—	4

Arithmetic Instructions							
Instruction	**Description**	**Flags**					**Obs.**
		CF	ZF	SF	IF	OF	
ADD dest, source	Adds source to dest	X	X	X	—	X	1
ADC dest, source	Adds source and CF to dest	X	X	X	—	X	1
SUB dest, source	Subtracts source from dest	X	X	X	—	X	1
SBB dest, source	Subtracts source and CF from dest	X	X	X	—	X	1
CMP dest, source	Compares source with dest	X	X	X	—	X	1
NEG dest	Negates dest	X	X	X	—	X	5
INC dest	Increments dest	X	X	X	—	X	5
DEC dest	Decrements dest	X	X	X	—	X	5

Logical Instructions							
Instruction	**Description**	**Flags**					**Obs.**
		CF	ZF	SF	IF	OF	
AND dest, source	Operation dest AND source bit-wise	0	X	X	—	0	1
OR dest, source	Operation dest OR source bit-wise	0	X	X	—	0	1
XOR dest, source	Operation dest XOR source bit-wise	0	X	X	—	0	1
TEST dest, source	Operation dest AND source bit-wise, flags only	0	X	X	—	0	1
NOT dest	Operation NOT dest bit-wise	0	X	X	—	0	5

Interrupt Handling Instructions							
Instruction	**Description**	**Flags**					**Obs.**
		CF	ZF	SF	IF	OF	
INT N	Executes software interrupt N	—	—	—	0	—	
IRET	Returns from the interrupt routine	X	X	X	X	X	
CLI	Disables maskable interrupts	X	X	X	0	X	
STI	Enables maskable interrupts	X	X	X	1	X	

Control Transfer Instructions							
Instruction	**Description**	**Flags**					**Obs.**
		CF	ZF	SF	IF	OF	
`CALL label`	Calls a subroutine starting at `label`	—	—	—	—	—	
`RET`	Returns from the subroutine	—	—	—	—	—	
`JC label`	Jumps to `label` if `CF` = `1`	—	—	—	—	—	
`JNC label`	Jumps to `label` if `CF` = `0`	—	—	—	—	—	
`JZ label`	Jumps to `label` if `ZF` = `1`	—	—	—	—	—	
`JNZ label`	Jumps to `label` if `ZF` = `0`	—	—	—	—	—	
`JS label`	Jumps to `label` if `SF` = `1`	—	—	—	—	—	
`JNS label`	Jumps to `label` if `SF` = `0`	—	—	—	—	—	
`JO label`	Jumps to `label` if `OF` = `1`	—	—	—	—	—	
`JNO label`	Jumps to `label` if `OF` = `0`	—	—	—	—	—	
`JMP label`	Unconditionally jumps to `label`	—	—	—	—	—	

Control Instructions							
Instruction	**Description**	**Flags**					**Obs.**
		CF	ZF	SF	IF	OF	
`NOP`	Does nothing	—	—	—	—	—	
`HLT`	Halts execution	—	—	—	—	—	

1. The possibilities for `dest, source` are:
   register, register; *register, memory address*; *register, immediate*; *memory address, register*; *memory address, immediate*.
   The *memory address* can be a label (direct addressing) or `[BX]`, which is a memory address (indirect addressing). It can also be an indirect addressing with offset in the form `[BX+offset]`.
2. `dest` and `source` can only be 16-bit registers.
3. The possibilities for `dest, source` are:
   - `AL`, *port*;
   - `AX`, *port*;
   - `AL`, `DX`;
   - `AX`, `DX`.

   port must be an immediate operand between `0` and `255`.
4. The possibilities for `dest, source` are:
   - *port*, `AL`;
   - *port*, `AX`;
   - `DX`, `AL`;
   - `DX`, `AX`.

   port must be an immediate operand between `0` and `255`.
5. `dest` can only be a *memory address* or a *register*. The *memory address* can be a label (direct addressing) or `[BX]`, which is a memory address (indirect addressing). It can also be an indirect addressing with offset in the form `[BX+offset]`.