OpCode	Mnemonic	Mode	Operand	Description
0	READ			reads a value from the input unit into register A
1	WRITE			writes the value from register A to the output unit
2	LOAD	A(0)	address	reads a value from the specified memory address into register A
2	LOAD	B(1)	address	reads a value from the specified memory address into register B
4	STORE	A(0)	address	writes the value from register A to the specified memory address
4	STORE	B(1)	address	writes the value from register B to the specified memory address
6	ADD	*REG(0)*		adds the values in A and B, storing the result back in A
7	ADDI	*A(0)*	#	adds the value in A with the operand, storing the result back in A
7	ADDI	*B(1)*	#	adds the value in B with the operand, storing the result back in B
8	SUB	*REG(0)*		subtracts the value in B from A, storing the result back in A
9	SUBI	*A(0)*	#	subtracts the operand from A, storing the result back in A
9	SUBI	*B(1)*	#	subtracts the operand from B, storing the result back in B
A	MUL	*REG(0)*		Multiplies the values in A and B, storing the result back in A
B	MULI	*A(0)*	#	multiplies the value in A by the operand, storing the result back in A
B	MULI	*B(1)*	#	multiplies the value in B by the operand, storing the result back in B
C	DIV	*REG(0)*		divides the value in A by B, storing the result back in A
D	DIVI	*A(0)*	#	divides the value in A by the operand, storing the result back in A
D	DIVI	*B(1)*	#	divides the value in B by the operand, storing the result back in B
E	MOD	*REG(0)*		remainder from dividing the value in A by B, storing the result back in A
F	MODI	*A(0)*	#	remainder from dividing the value in A by the operand, storing the result back in A
F	MODI	*B(1)*	#	remainder from dividing the value in B by the operand, storing the result back in B
15	JUMPIFNOTZERO		*address*	loads PC with the specified address, if A does not contain 0
16	JUMPIFZERO		*address*	loads PC with the specified address, if A contains 0
17	JUMP		*address*	loads PC with the specified address
18	HALT			halts the CPU, no further instructions will be executed (until reset)
19	RETURN			Returns out of a method
1A	CALL		*address*	Calls a method
1B	INCSP		*offset*	adds the specified offset to SP. A negative offset allocates stack space (pushes); a positive value deallocates it (pops)
1C	LOADREL	*A(0)*	*offset*	reads a value from the specified memory address (SP + offset) into register A
1C	LOADREL	*B(1)*	*offset*	reads a value from the specified memory address (SP + offset) into register B
1D	STOREREL	*A(0)*	*offset*	writes the value from register A to the specified memory address (SP + offset)
1D	STOREREL	*B(1)*	*offset*	writes the value from register B to the specified memory address (SP + offset)