Befehl	Argument(e)	Beschreibung
add	–	Addiert die zwei obersten Elemente des Stacks und entfernt diese. Das Ergebnis wird auf den Stack gelegt.
and	–	Verknüpft die zwei obersten Elemente mit der AND-Operation und entfernt diese. Das Ergebnis wird auf den Stack gelegt.
dec	–	Dekrementiert den obersten Wert auf dem Stack.
inc	–	Inkrementiert den obersten Wert auf dem Stack.
or	–	Verknüpft die zwei obersten Elemente mit der OR-Operation und entfernt diese. Das Ergebnis wird auf den Stack gelegt.
nop	–	Keine Operation.
sub	–	Subtrahiert die zwei obersten Elemente des Stacks und entfernt diese. Das Ergebnis wird auf den Stack gelegt. Dabei wird das zweite Element vom obersten Element subtrahiert.
swap	–	Vertauscht die zwei obersten Elemente auf dem Stack.
xor	–	Verknüpft die zwei obersten Elemente mit der XOR-Operation und entfernt diese. Das Ergebnis wird auf den Stack gelegt.
call	<addr>	(Funktionsaufruf) Springt an die Stelle von <addr> und führt den dort stehenden Code aus, bis ein return erreicht wird.
return	–	Springt zurück an die Stelle des letzten Funktionsaufrufes.
goto	<addr>	Springt bedingungslos an die Stelle <addr>.
jmz	<addr>	Springt an die Stelle <addr>, falls das oberste Element des Stacks gleich null ist.
jmnz	<addr>	Springt an die Stelle <addr>, falls das oberste Element des Stacks ungleich null ist.
jmc	<addr>	Springt an die Stelle <addr>, falls die obersten zwei Stackelemente gleich sind.
pop	–	Entfernt das oberste Element vom Stack.
push	<element>	Legt den Wert <element> auf den Stack.
shl8	–	Führt eine Schiebeoperation um 8 Bits nach links auf dem obersten Stackelement aus.
shr1	–	Führt eine Schiebeoperation um 1 Bit nach rechts auf dem obersten Stackelement aus.
dup	–	Erstellt eine Kopie des obersten Stackelements und legt diese auf den Stack.
load8	<addr>	Lädt einen 8-Bit Wert aus dem RAM an der Adresse <addr> und legt diesen auf den Stack.
load16	<addr>	Lädt einen 16-Bit Wert aus dem RAM an der Adresse <addr> und legt diesen auf den Stack.
store	<addr>	Speichert das oberste Stackelement im RAM an der Adresse <addr>.

Auf Wunsch implementieren wir auch noch weitere Befehle (z.B. load32).