

Assemblercode

Adressierung: *Register, [Register], [Adresse], Speicher-/Sprungziele*

Beispiel: [AX] (*Adresse, die im Register AX steht*)

Adressbereich (&H): 00 - FF

Register: AX (Akkumulator), BX, CX, DX

Sprungziele: *:Name*

Speicherziele: *var Name*

(reserviert Speicher, an dem *var Name* stehen würde, als Zahlenspeicher, ansprechbar mit *Name*)

Zahlen: *Hexadezimal-Zahl (&H)*

Beispiel: 3A

Befehle:

Par1, Par2 sind Adressen oder Zahlen, sofern sinnvoll

Zero-Flag: Wird gesetzt (1), wenn die vorhergehende Operation als Ergebnis 0 hat

nop	Befehl ohne Wirkung
add <i>Par1, Par2</i>	Ergebnis der Summe <i>Par1+Par2</i> in <i>Par1</i> gespeichert
sub <i>Par1, Par2</i>	Ergebnis der Differenz <i>Par1-Par2</i> in <i>Par1</i> gespeichert
and <i>Par1, Par2</i>	Ergebnis der Konjunktion <i>Par1</i> and <i>Par2</i> in <i>Par1</i> gespeichert
or <i>Par1, Par2</i>	Ergebnis der Disjunktion <i>Par1</i> or <i>Par2</i> in <i>Par1</i> gespeichert
not <i>Par1</i>	Ergebnis der Negation not <i>Par1</i> in <i>Par1</i> gespeichert
cmp <i>Par1, Par2</i>	Vergleich <i>Par1, Par2</i> , Ergebnis in <i>Zero-Flag</i> gespeichert
shl <i>Par1</i>	Verschieben von <i>Par1</i> um ein Bit nach links
mov <i>Par1, Par2</i>	Kopieren von <i>Par2</i> in Speicher <i>Par1</i>
push <i>Par1</i>	Kopieren von <i>Par1</i> auf den Stack
pop <i>Par1</i>	Auslesen des letzten Stackeintrags in <i>Par1</i>
jmp <i>Par1</i>	Sprung nach Adresse <i>Par1</i>
jz <i>Par1</i>	Sprung nach Adresse <i>Par1</i> , wenn das <i>Zero-Flag</i> gesetzt ist
jnz <i>Par1</i>	Sprung nach Adresse <i>Par1</i> , wenn das <i>Zero-Flag</i> nicht gesetzt ist
end	Pseudobefehl: Programmende