Vom Quellcode zum Prozessor

1. **Compiler übersetzt Quellcode in Maschinensprache**
   -> Compiler muss wissen welche CPU weil unterschiedlich Sprache
   -> z.B: 16 Bit CPU = Befehle von 16 Bit können überführt werden

void main (void){
  int x = 2;
  int y = 5;
  int z = x + y;
}
	zu	0000000011000010000000010000110100

2. **Maschinensprache in Einzelteile zerlegen**
   -> Wenn Nummer = 1, dann geht es um die Zahl, sonst um die Speicherzelle
   -> die Befehle werden auf dem ACC durchgeführt

Reserve	Befehl	Nr.	Operant	Bedeutung		Nr.	Befehl	Bedeutung

000000 000000 000000 000000 000000 000000 000000 000000	001 010 001 010 001 011 010 111	1 0 1 0 0 0 0 0	000010 001101 000101 001110 001101 001110 001111 000000	Ladeden Wert 2in ACC Speichere ACC in Speicherzelle 13 Ladeden Wert 5in ACC Speichere ACC in Speicherzelle 14 Lade Speicherzelle 13 inACC Addiere Speicherzelle in 14 zu ACC Speichere ACC in Speicherzelle 15 Programm beenden		000 001 010 011 100 101 110 111	NOOP LOAD STORE ADD SUB EQUAL JUMP HALT	No Operation Was laden Speicher Addieren Minimieren Vergleichen Überspringe Beenden

---

Befehlsformat

> Das **Befehlsformat** definiert für jeden einzelnen Befehl, wie dieser codiert ist: <Befehl><Num><Operand>

> **Opcode -** Binäre Codierung, aus der sowohl der Befehl, als auch zusätzlich benötigte Steuerinformationen hervorgehen.

Klassifizierungen - Bei den Befehlsformaten werden verschiedene Klassifizierungen unterschieden:

- Einadressformat | Zweiadressformat | Dreiadressformat

> Das **Einadressformat** entspricht dem Format Vom Quellcode zum Prozessor. Hier wird für die einzelnen Befehle nur der *Opcode* und die *Adresse eines Operanden* (deshalb *Einadressformat*) angegeben.

<Opcode><Operand1>

Bedeutung des Akkumulators

Wird ein zweiter **Operand** benötigt, so wird vorausgesetzt, dass dieser **sich im Register Akkumulator** befindet. Das bei der Abarbeitung des Befehls berechnete *Ergebnis* wird per Definition wieder im *Akkumulator* gespeichert.

Revision #3
Created 18 August 2022 08:48:05 by Merith Holtmann
Updated 1 October 2022 08:32:24 by Merith Holtmann