

3. AUFLAGE HERAUSGEGEBEN VON ARNDT BODE,
WOLFGANG KARL UND THEO UNGERER

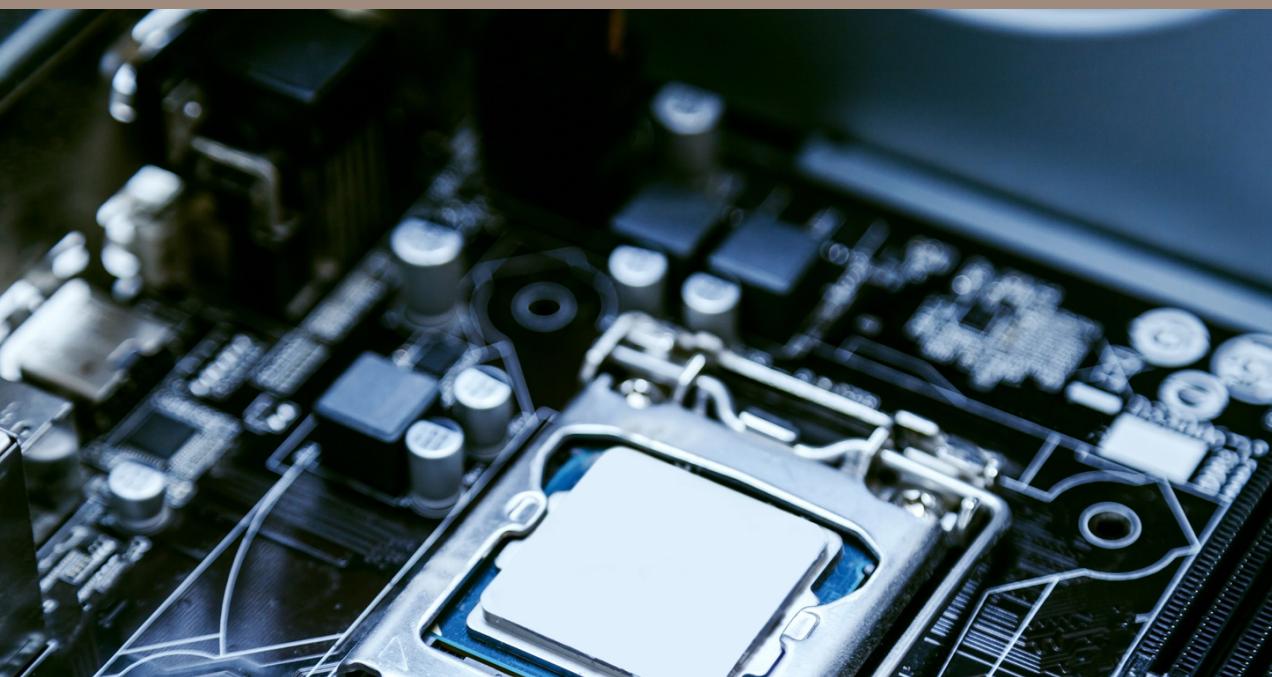
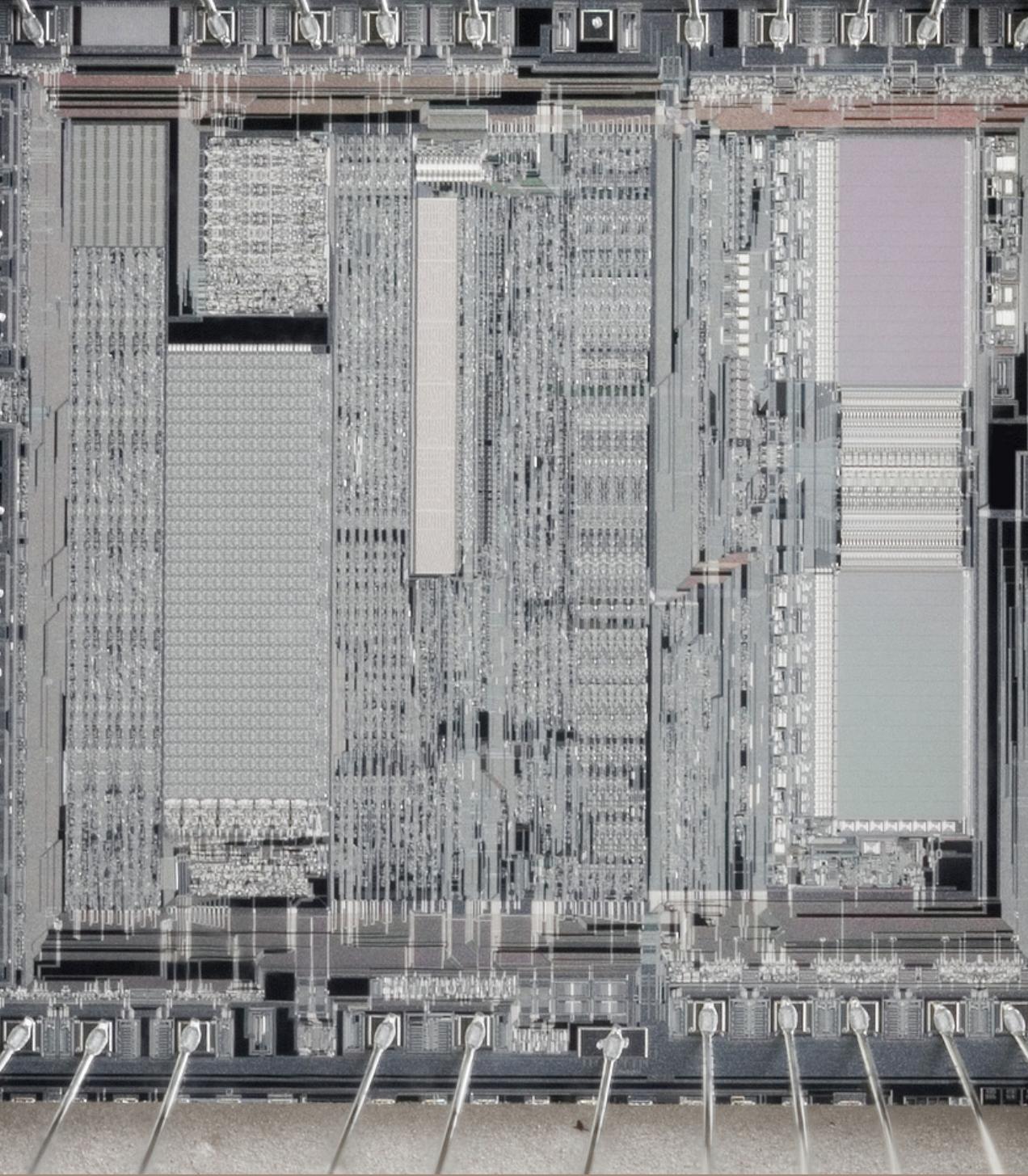
RECHNERORGANISATION UND -ENTWURF

Befehle zum Treffen von Entscheidungen

2.6 Befehle zum Treffen von Entscheidungen

bne register1, register2, L1

Übersetzung einer if-then-else-Anweisung in eine bedingte Verzweigung



Schleifen

Hardware-
Software-
Schnittstelle



CASE- / SWITCH ANWEISUNG

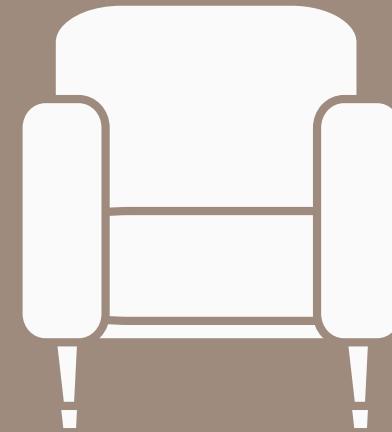
sprungadresstabelle MIPS-Architektur



32 Register



2^{30}
Speicherwörter

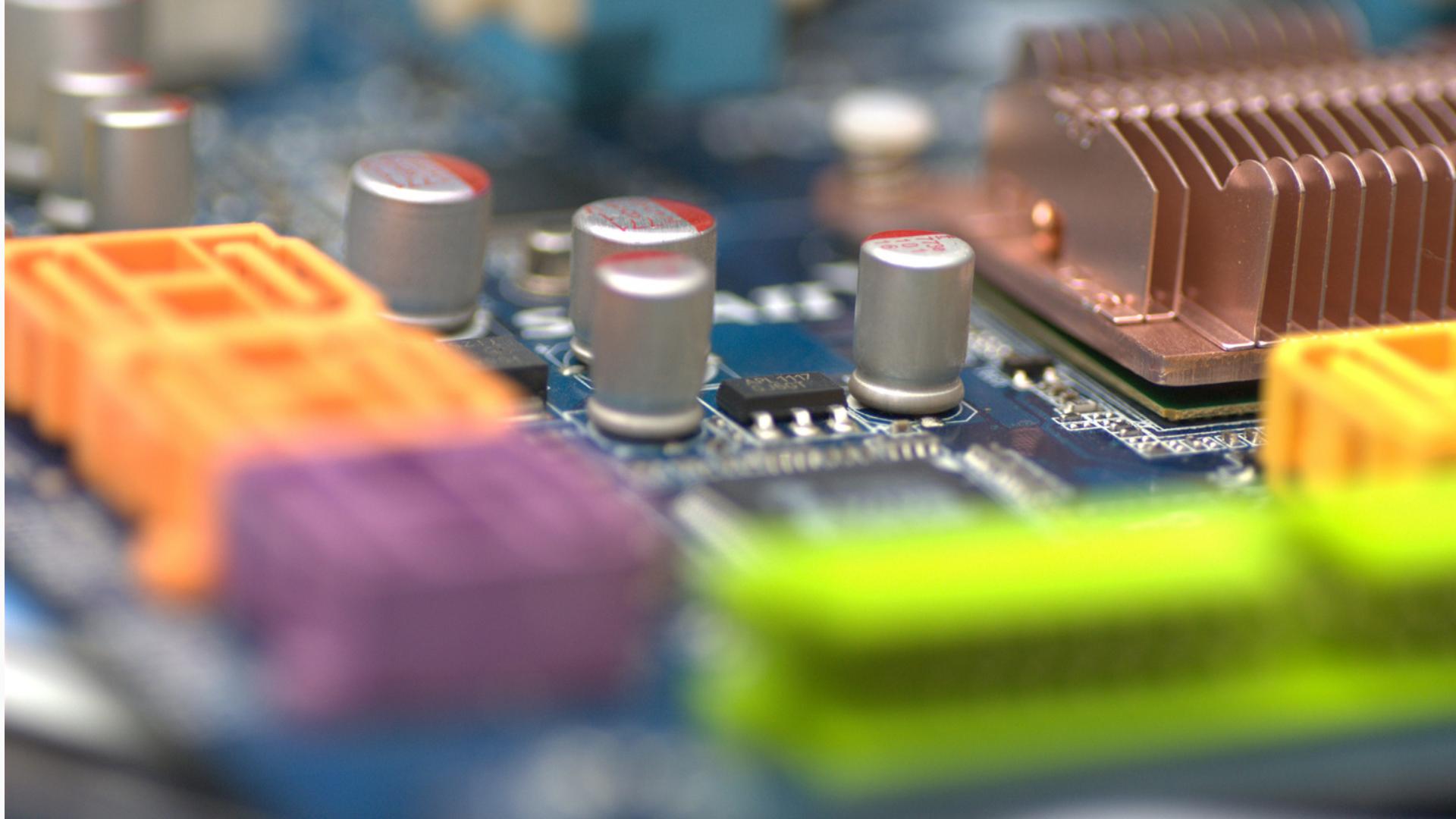


Arithmetisches Befehl

DIE OBIGE AUSSAGE GILT GENAU UMGEGEHRT: && UND II
ENTSPRECHEN LOGISCHEN OPERATIONEN, & UND I BEDINGTEN
VERZWINGUNGEN.

2.7 Unterstützung von Prozeduren durch die Rechnerhardware

Prozedur: Verwendung weitere Register



2.7 BEFEHLE: DIE SPRACHE DES RECHNERS

Prozessor, Hauptspeicher

BESPIEL
ANTWORT



2.7 Geschachtelte Prozeduren

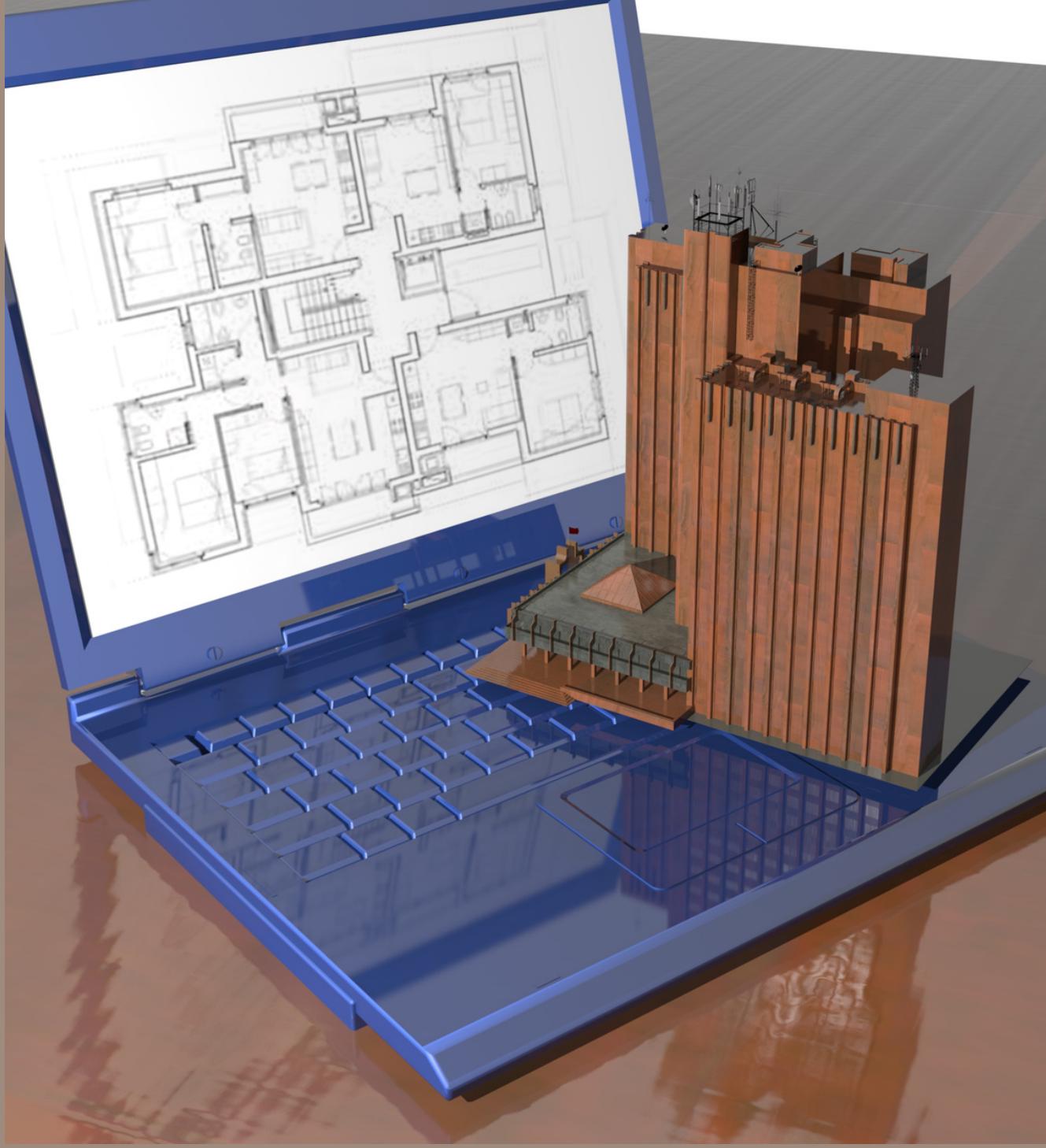
```
int fact (int  
{  
if (n < 1 ) return (1);  
else return (n * fact(n-1));  
}
```

Beispiel

Antwort

Zuordnen von Speicherplatz für neue Daten im Keller

Prozeduraufrufrahmen



2.8

Kommunikation mit Menschen

MIPS-Maschinensprache

Zeichen und Zeichenfolgen in Java

UNICODE

DIE SPRACHE DES RECHNERS

Umgang mit 32-Bit Direktoperanden und 32-Bit- Adressen

32-bit-Direktoperanden

Beispiel

welche der folgenden Aussagen über Zeichen und Zeichenfolgen in C und Java treffen zu?



Antwort

Eine Zeichenfolge in C belegt etwas so viel Speicherplatz wie dieselbe Zeichenfolge in Java.

Adressbildung bei Verzweigungen und Sprüngen

j 10000 # springe an Stelle 10000

Befehlszähler = Register + Sprungadresse



MIPS-Adressierungsarten

eine Übersicht

1. Registeradressierung
2. Basis - Displacement-Adressierung
3. Direkte Adressierung
4. Befehlszählerralitive Adressierung
5. Pseudo-direkte Adressierung



Entschlüsseln der Maschinensprache

MIPS-Befehlscodierung

Entschlüsseln des Maschinencodes

2.9 UMGANG MIT 32-BIT DIREKTOPERANDEN UND 32-BIT
ADRESSEN



FRAGEN

Wie lautet der Adressbereich für bedingte Sprünge bei MIPS ($K = 1024$) ?

Adressen zwischen 0 und $64 K - 1$

FRAGEN

Wie lautet der Adressbereich für Sprüngh- und Jump-and-Link-Befehle bei MIPS ($M = 1024 \text{ K}$)?

Adressen zwischen 0 und $64 M - 1$

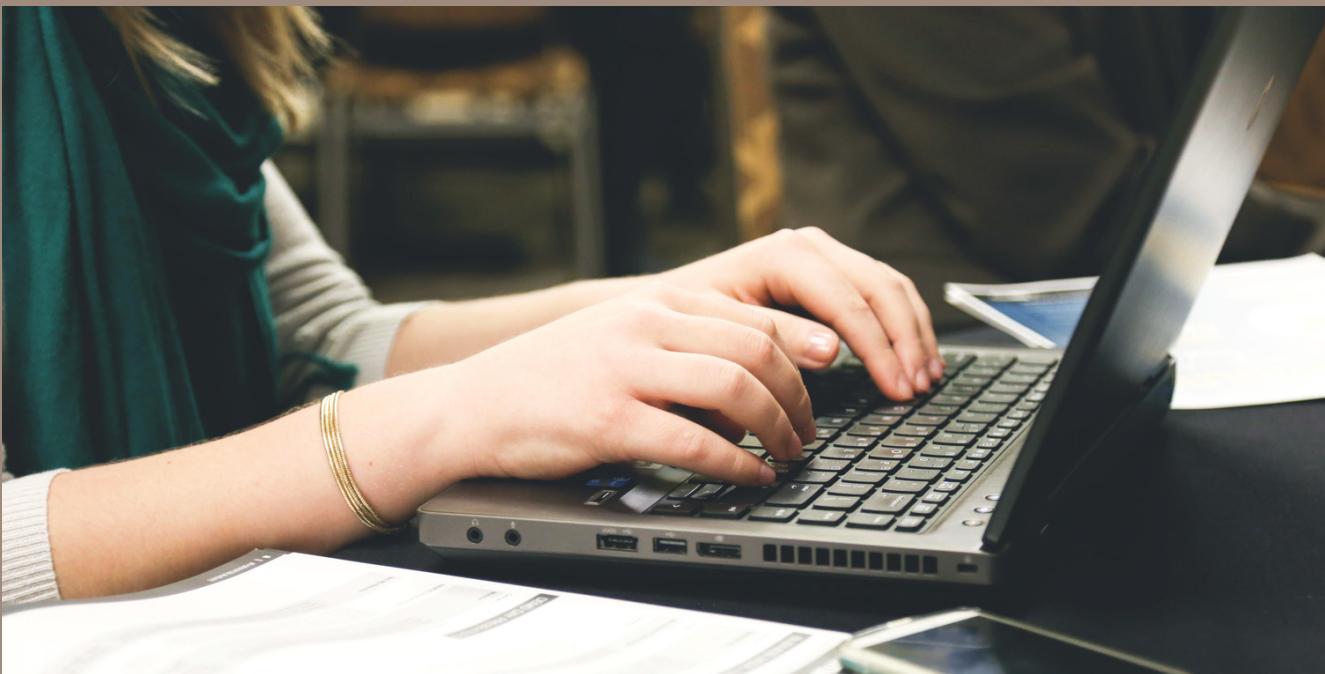
2.10 übersetzen und Starten eines Programms

Eine Übersetzungshierarchie für C.

Compiler

COMPILER

```
$(window).scrollTop() > header1_init  
if (parseInt(header1.css('padding-top')) > header1_init)  
    header1.css('padding-top', '')  
} else {  
    header1.css('padding-top', '' + header1_top)  
  
$(window).scrollTop() > header2_init  
if (parseInt(header2.css('padding-top')) > header2_init)  
    header2.css('padding-top', '')  
} else {  
    header2.css('padding-top', '' + header2_top)
```



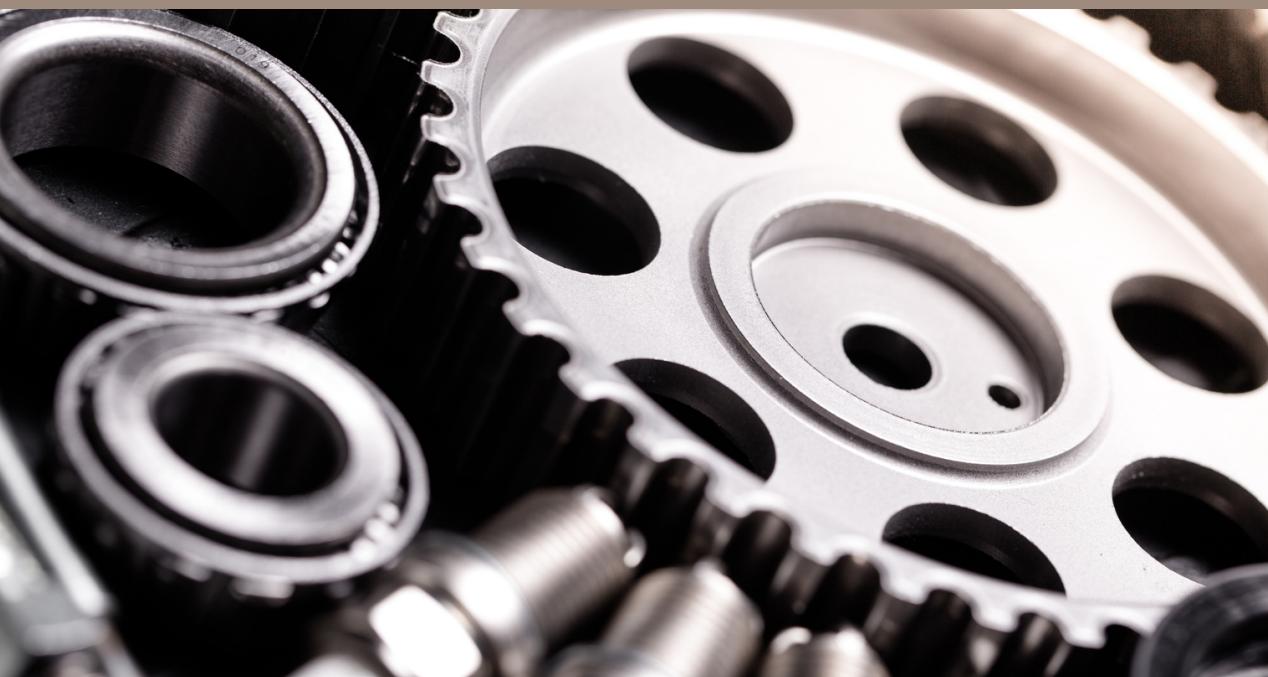
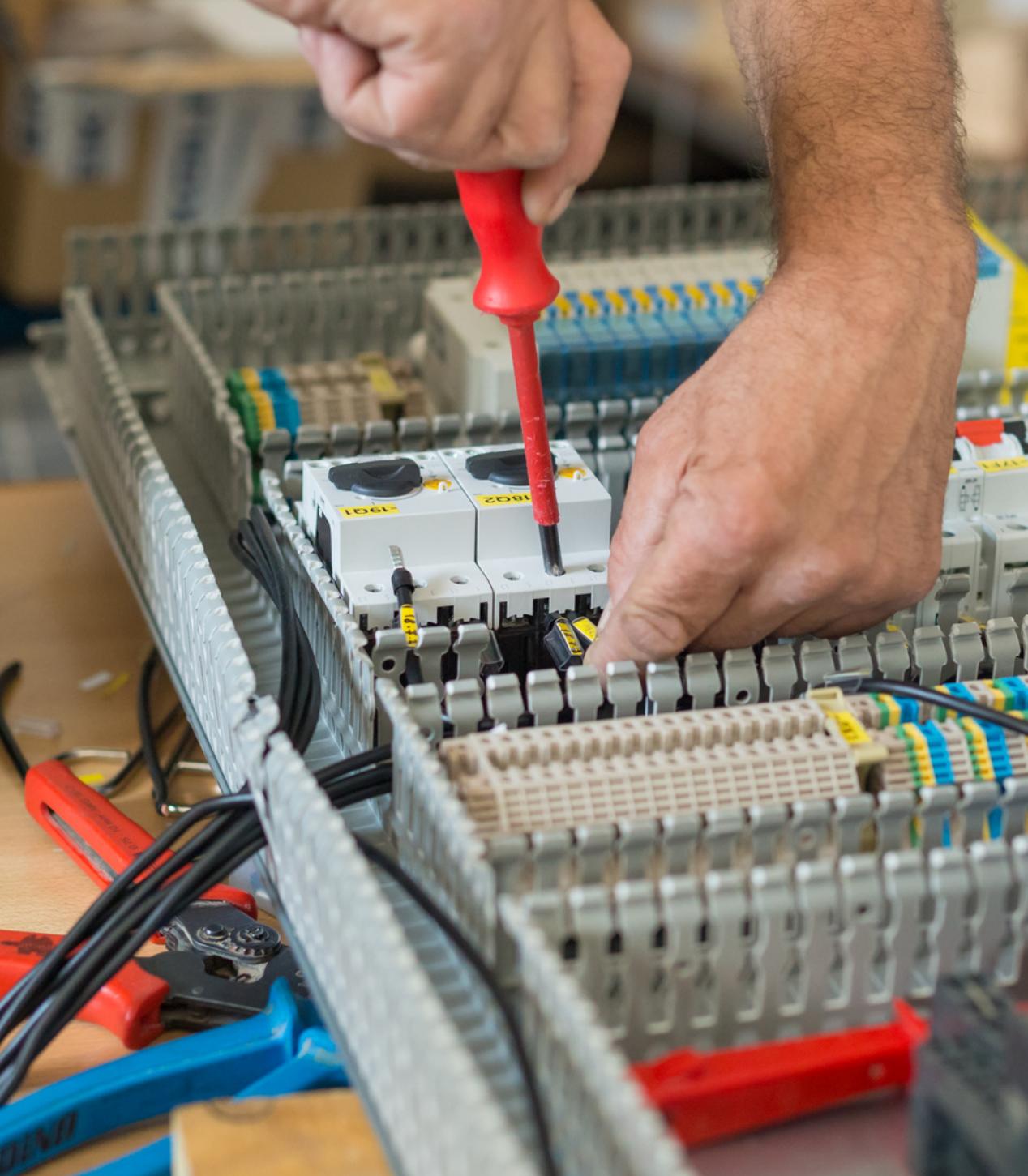
Assembler

Pseudobefehle(pseudoinstruction)

Eine allgemeine Variation von Befehlen in der Assembler Sprache, die wie ein tatsächlicher Befehl behandelt wird.

Maschinensprache

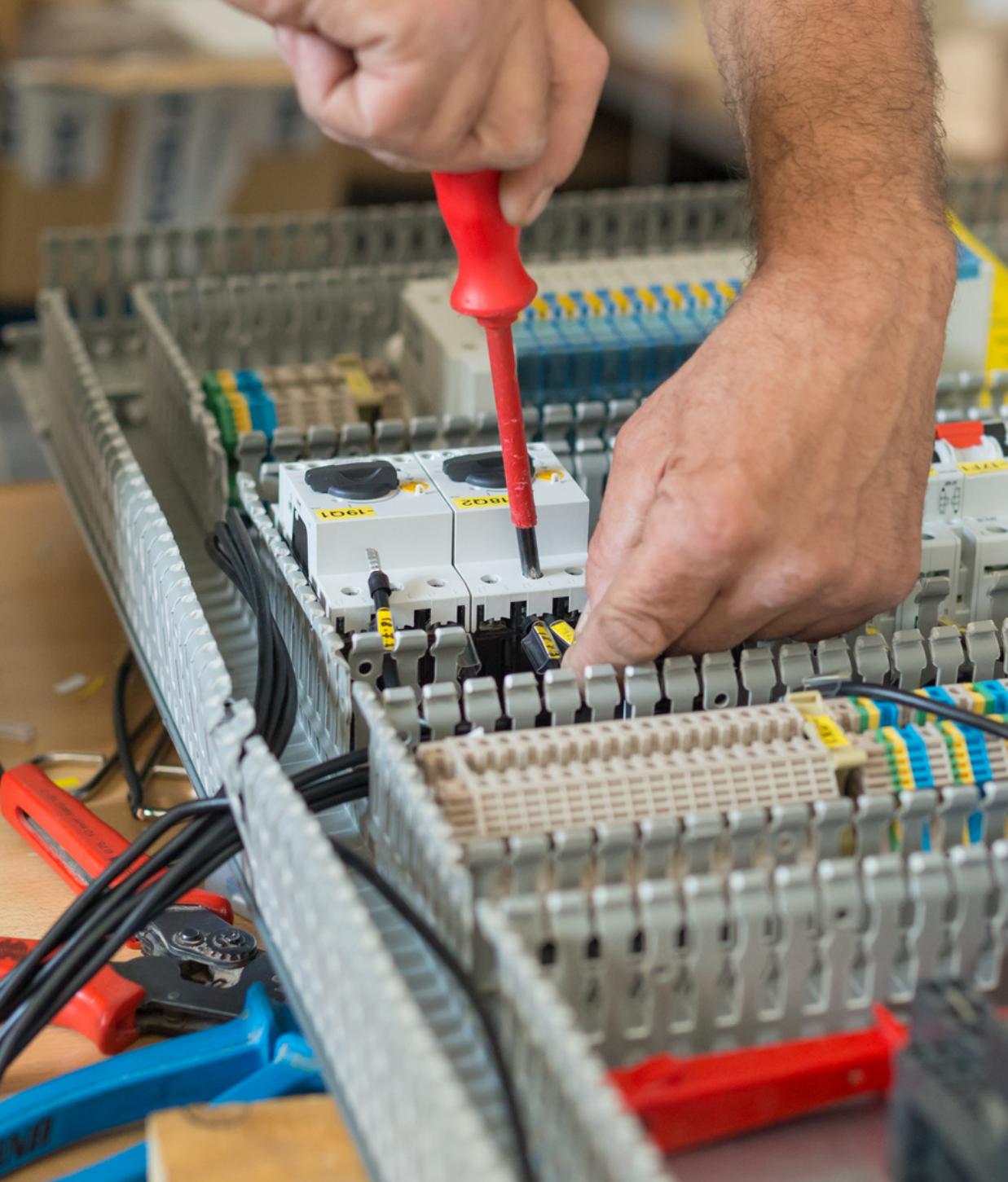
Symboltabelle



Binder (linker)

Objektdateien binden

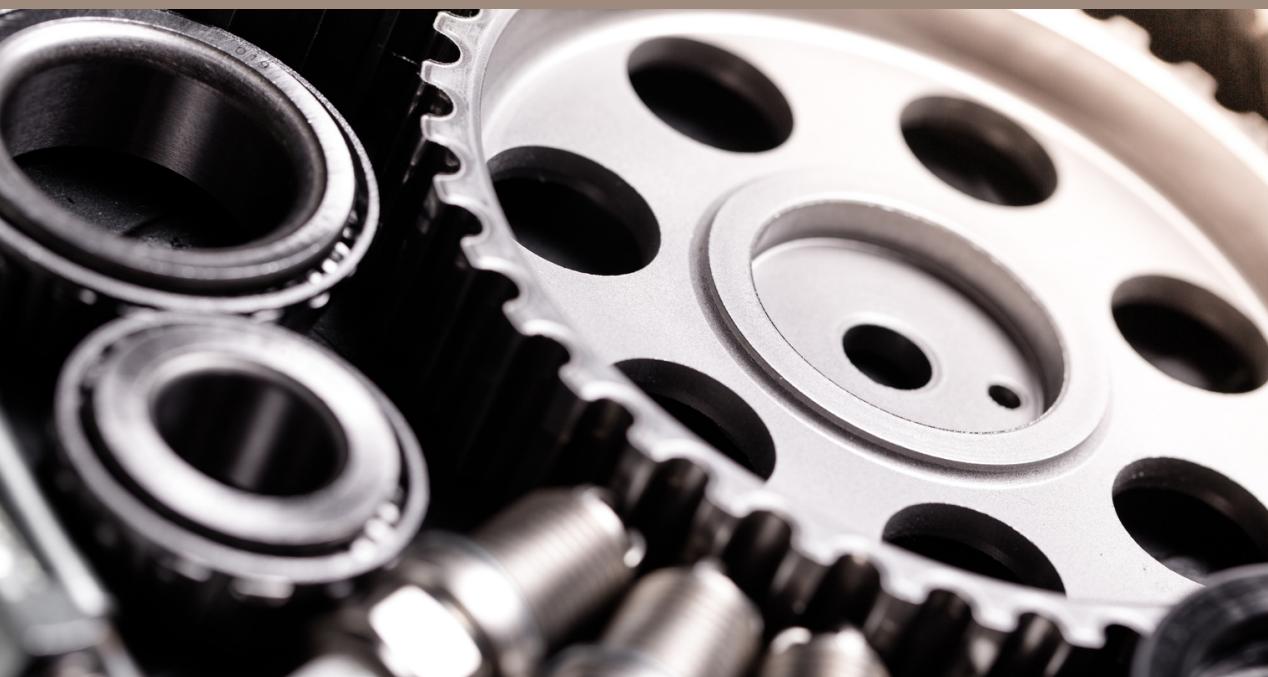
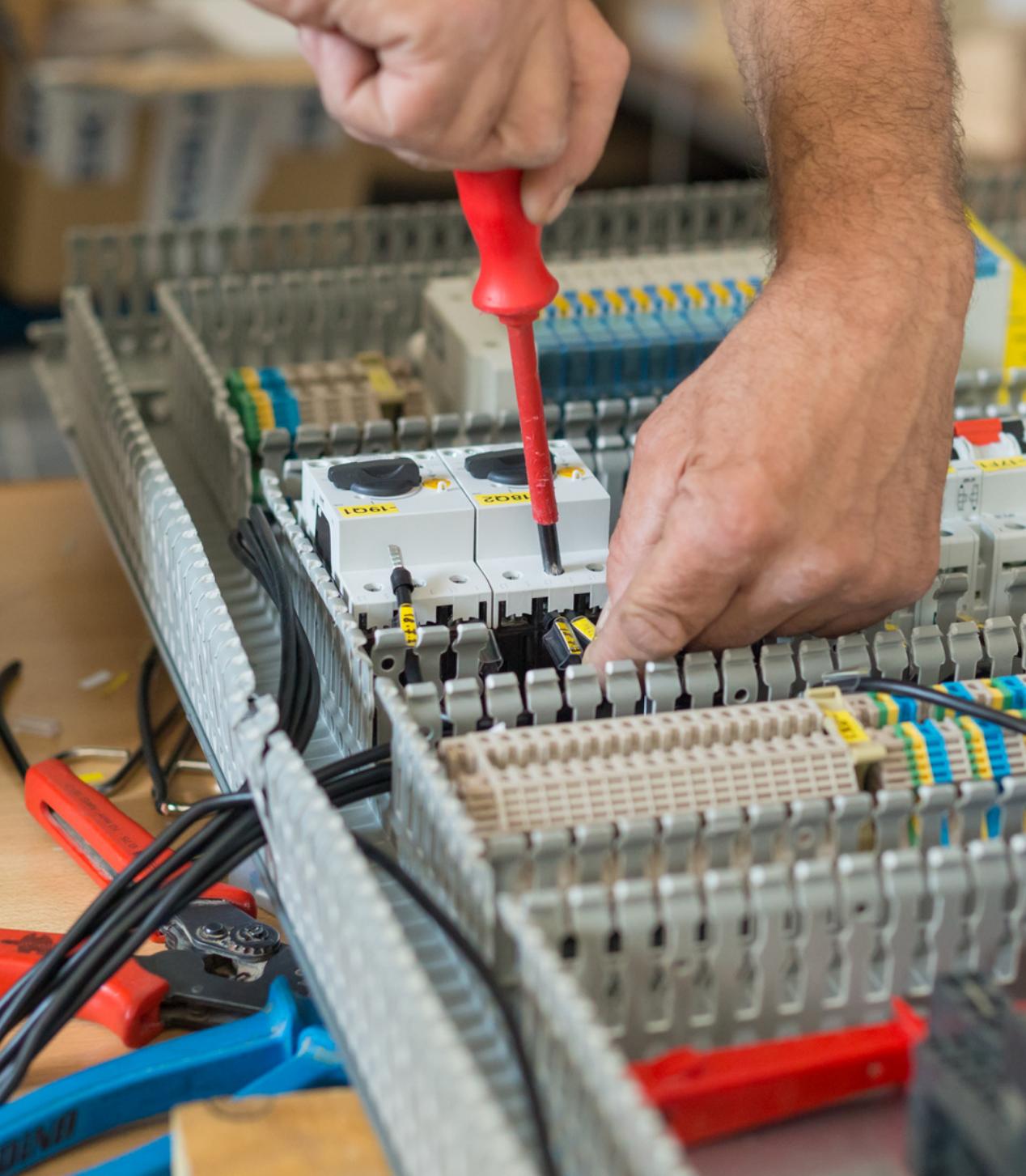
Header der ausführbaren Programmdatei



Lader

Dynamisch gebundene Bibliotheken (DLLs,
Dynamically linked Libraries)

Header der ausführbaren Programmdatei

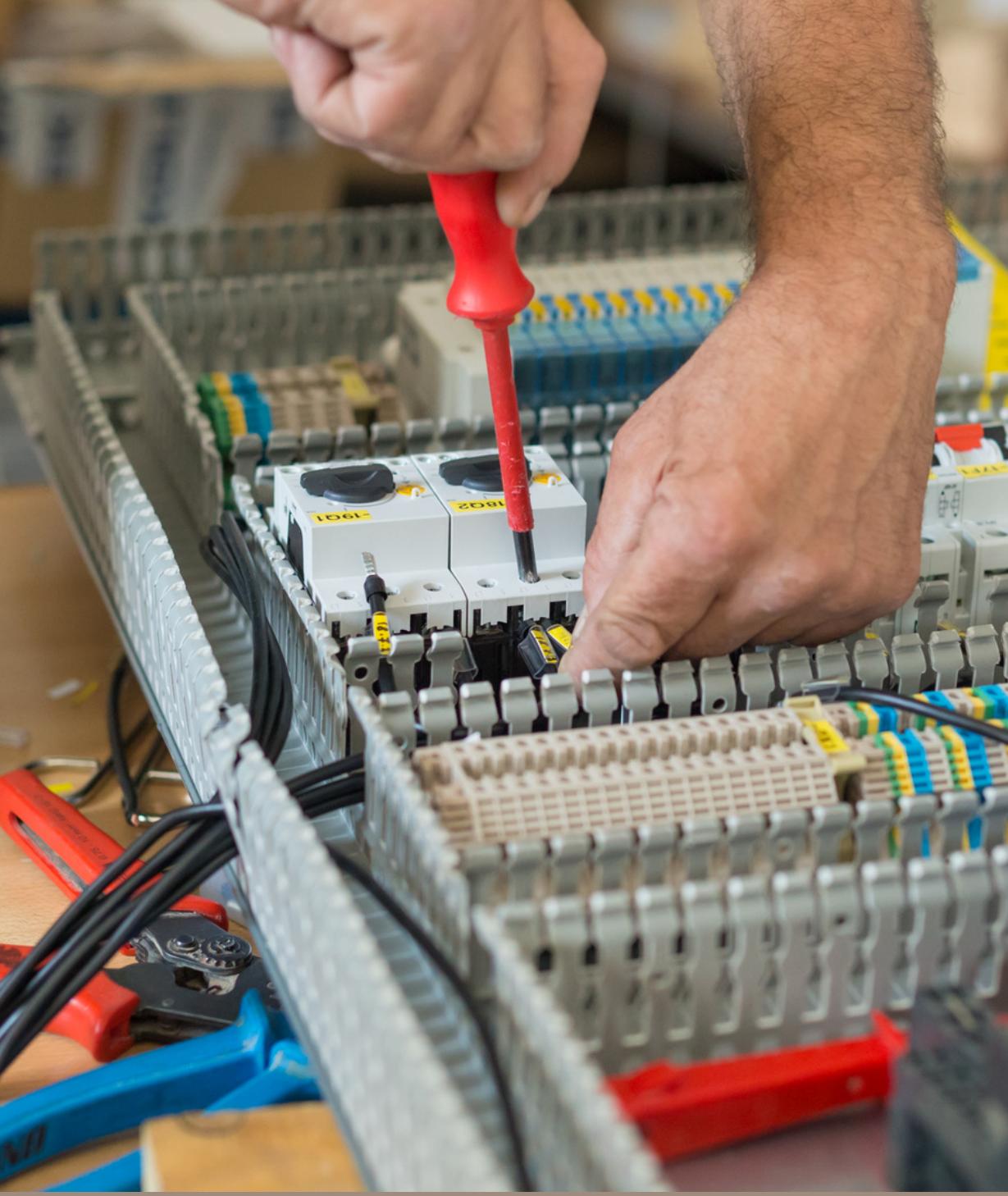


Starten eines Java-programms

java-bytecode

Java Virtual Machine

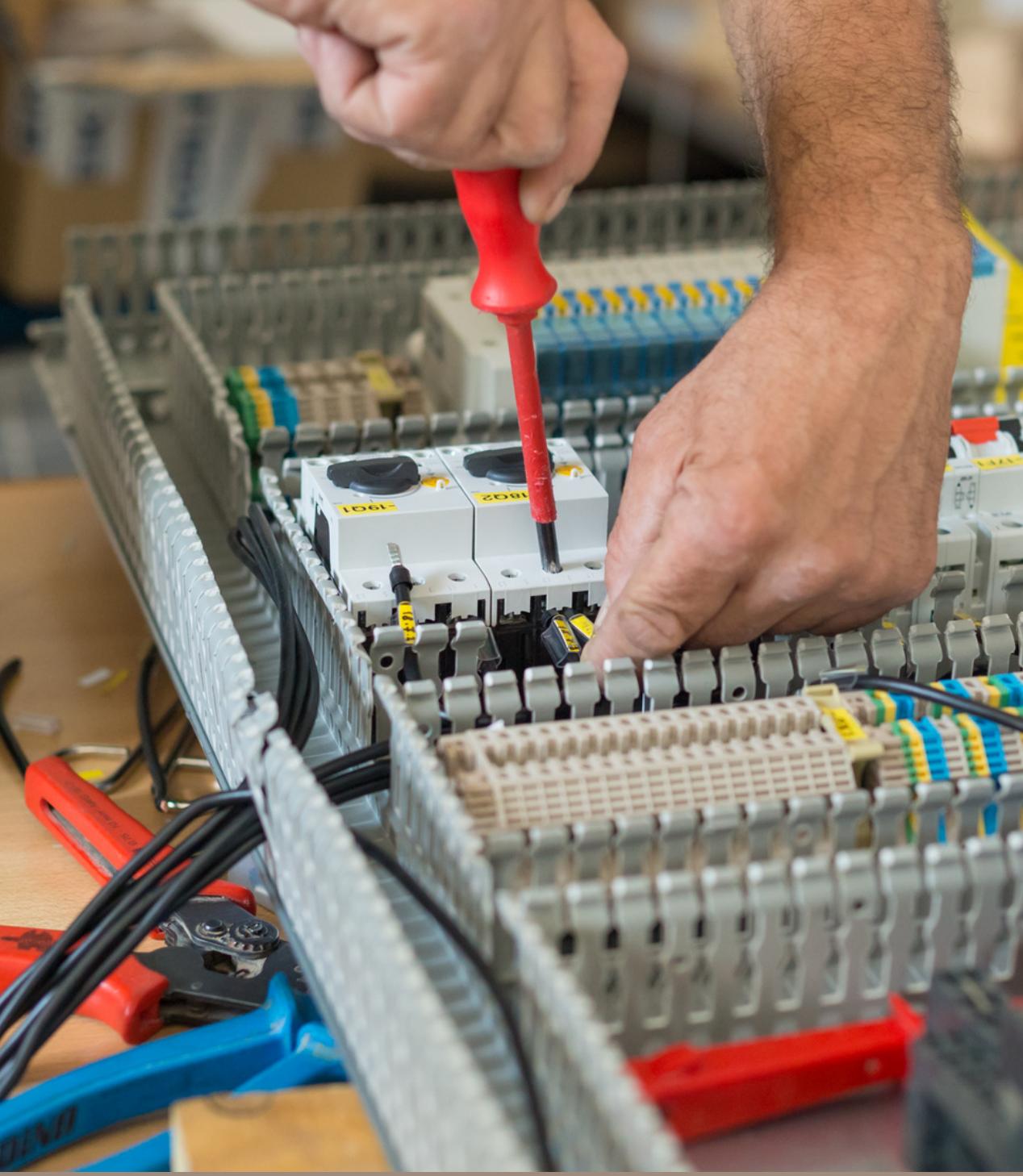
COMPILER



Optimierung durch den Compiler

Optimierungen auf den höchsten Stufe

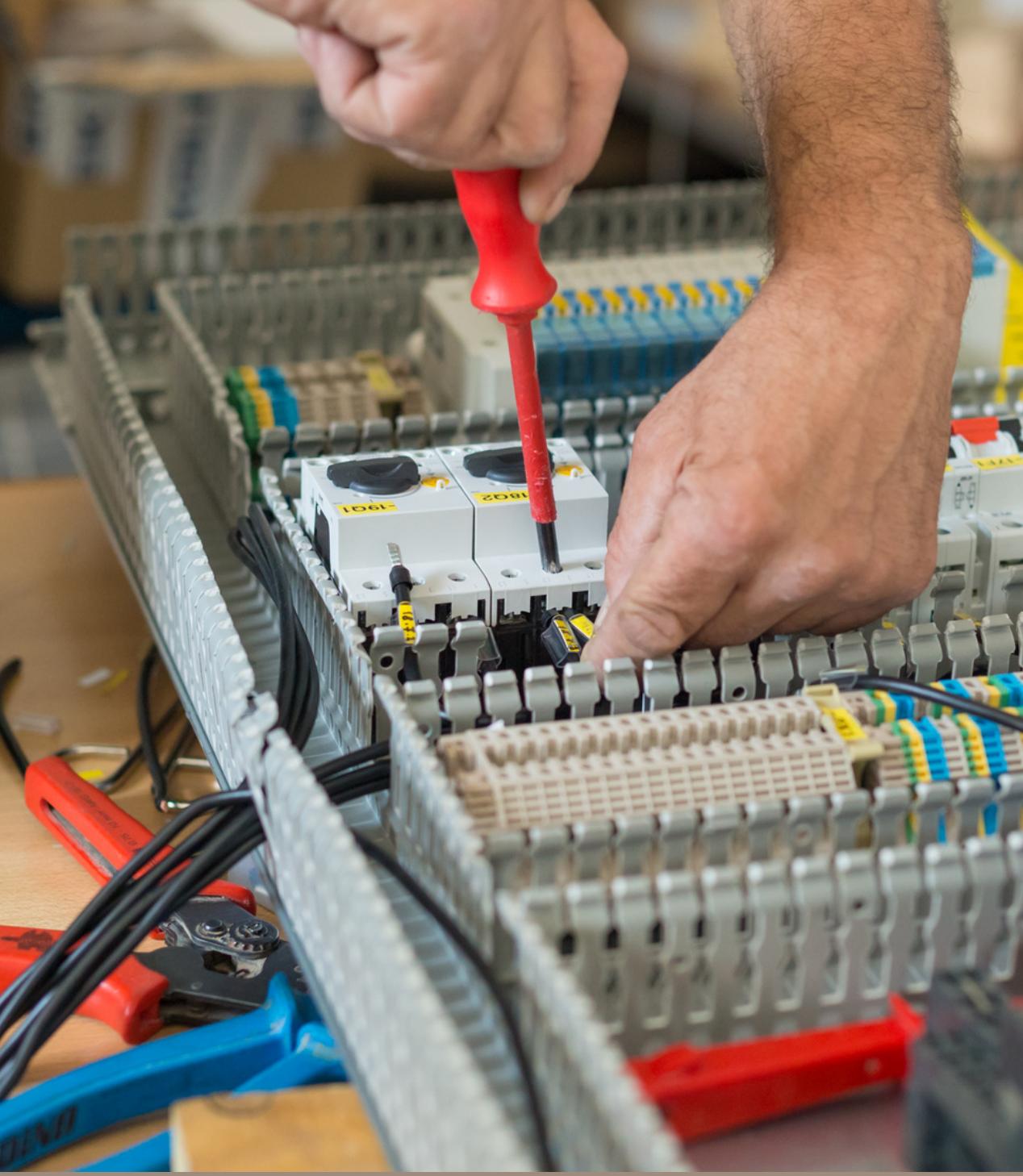
COMPILER



Grundlegendes zur Leistungsfähigkeit von Programmen

Globale Codeoptimierungen

COMPILER



2.12

Wie arbeiten Compiler: Eine Einführung

2.12

Zusammenfassung am Beispiel eines Sortierprogramms in C

Die Prozedur swap-prozedur

Die vollständige Prozedur swap

Dir Prozedur sort

```
Erweiterung: Docker | TS page-worker.ts 4 ×
src > TS page-worker.ts > ...
6
7  const flag = process.env.FLAG ?? "flag{missing}"
8
9  const sleep = (ms: number) => new Promise((resolve) => setTimeout(resolve, ms));
10
11 const visitOne = async () => {
12   const url = await dequeue();
13   if (url === undefined) {
14     await sleep(500);
15     return;
16   }
17
18   const browser = await puppeteer.launch({
19     dumpio: true,
20     pipe: true,
21     args: ['--no-sandbox', '--disable-setuid-sandbox']
22   });
23   const page = await browser.newPage();
24   await page.setCookie({
25     name: "flag",
26     value: flag,
27     domain: "localhost:3838",
28   })
29
30   await Promise.race([
31     page.goto(url),
32     sleep(3000),
33   ]);
34   await sleep(3000);
35
36   await browser.close();
37 }
38
39 export const startVisiting = async () => {
40   while (true) {
41     try {
42       await visitOne();
43     } catch (e) {
44       console.error(e);
45     }
46   }
47 }
48
49
```

Der Prozeduraufruf in sort

Übergabe von Parametern in sort

beibehalten von Registern in sort

Die vollständige Prozedur sort

TS database.ts 2 ×

src > TS database.ts > ...
44 | FROM queue;
45 | , [])
46 |
47 | return queueLength;
48 }
49
50 export const dequeue = async () => {
51 const request = await run<{ url: string, ip: string }>(

52 | SELECT url, ip
53 | FROM queue
54 | ORDER BY id ASC
55 | LIMIT 1;
56 | , []);
57
58 if (request.length === 0) {
59 | return undefined;
60 }
61
62 const [{ url, ip }] = request;

63
64 // Delete based off of url and ip in case there are
65 await run(`
66 | DELETE FROM queue
67 | WHERE url = ? AND ip = ?;
68 | , [url, ip]);
69
70 return url;
71 }

Grundlegendes zur Leistungsfähigkeit von Programmen

Leistungsverluste zweier Sortieralgorithmen in C und Java mit Interpretation und optimierenden Compilern im Vergleich zu nicht optimierter C-version.

Dockerfile X

Dockerfile

```
1 FROM node:lts
2
3 RUN apt-get update && apt-get install -
4
5 WORKDIR /problem
6 ADD .yarnrc.yml .
7 ADD .yarn ./yarn/
8 ADD package.json .
9 ADD yarn.lock .
10
11 RUN yarn
12
13 ADD . .
14 RUN yarn build
15
16 CMD ["yarn", "start"]
17
```