

Rechnertechnik

Prof. Dr. rer. nat. Peter Gerwinski

7. Juni 2021

Rechnertechnik

- 1 Einführung**
- 2 Vom Schaltkreis zum Computer**
- 3 Architekturmerkmale von Prozessoren**
- 4 Der CPU-Stack**
 - 4.1 Implementation
 - 4.2 Unterprogramme
 - 4.3 Register sichern

...

4 Der CPU-Stack

4.1 Implementation

Speicher, in dem Werte „gestapelt“ werden: *Stack*

- Speicherbereich (ein array) reservieren
- Variable (typischerweise: Prozessorregister) als *Stack Pointer* reservieren
→ *SP*
- Assembler-Befehl *push foo*: $*SP++ = foo;$
- Assembler-Befehl *pop bar*: $bar = *--SP;$

Speziell: Unterprogramme

4 Der CPU-Stack

4.2 Unterprogramme

Parameter:

- Prozessorregister
- CPU-Stack

Rückgabewert:

- Prozessorregister

Aufruf:

- push IP
 jmp foo ← mov #foo IP
 → call foo

Rücksprung:

- pop IP
 → ret

4 Der CPU-Stack

4.3 Register sichern

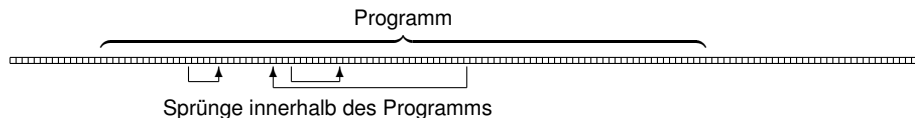
Ein Unterprogramm verändert Registerinhalte.

- im Hauptprogramm nötigenfalls vor Aufruf sichern
- im Unterprogramm vor Benutzung sichern
- Kombinationen (manche Register so, manche so)

5 Anwender-Software

5.1 Relokation und Linken

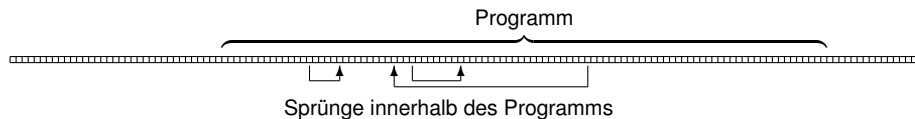
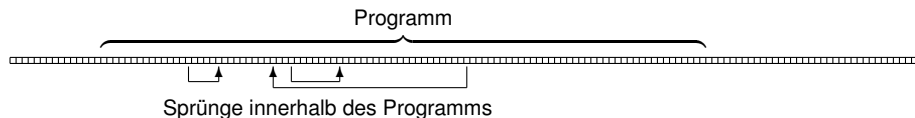
Software im Speicher



5 Anwender-Software

5.1 Relokation und Linken

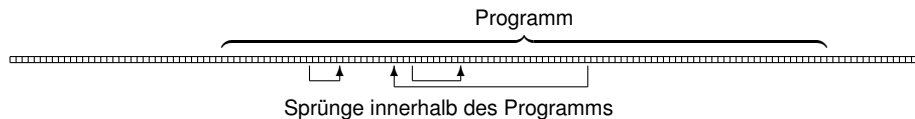
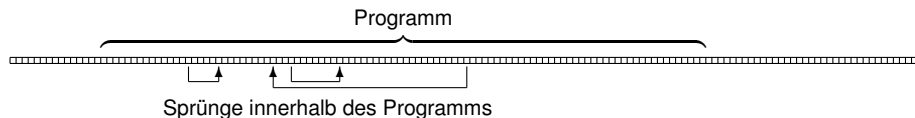
Software im Speicher



5 Anwender-Software

5.1 Relokation und Linken

Software im Speicher

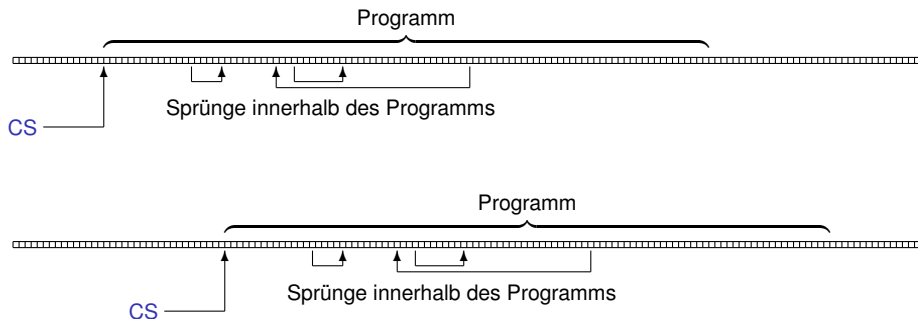


Sprünge anpassen: Relokation

5 Anwender-Software

5.1 Relokation und Linken

Software im Speicher

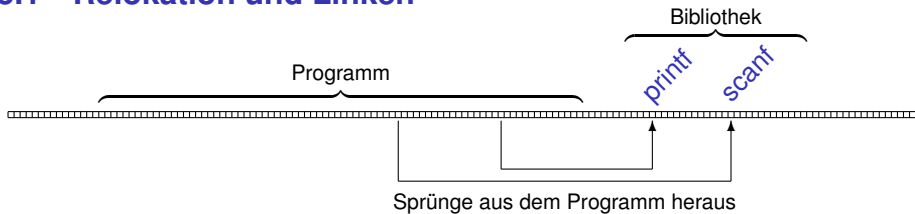


Sprünge anpassen: Relokation

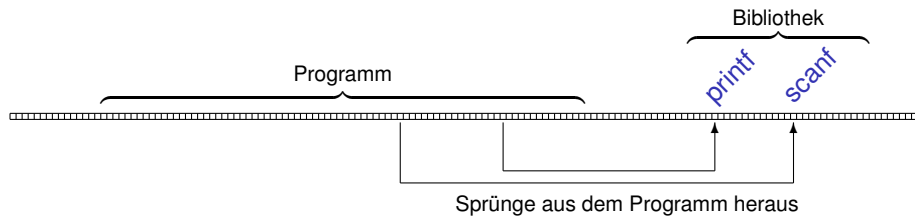
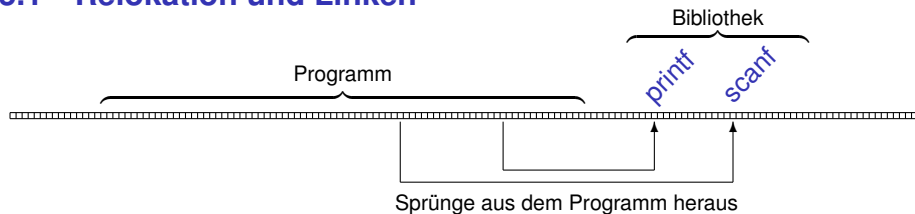
Hardware-Unterstützung (z. B. Intel): Speichersegmentierung

CS = Code-Segment: Segment-Register oder Selektor

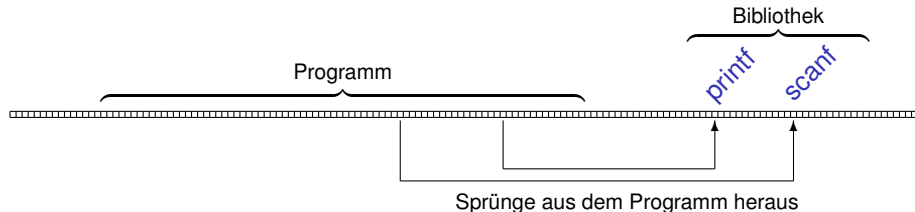
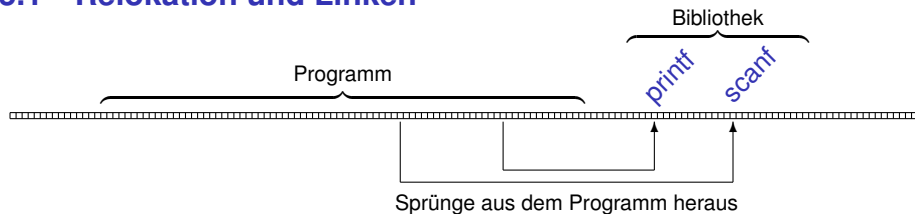
5.1 Relokation und Linken



5.1 Relokation und Linken

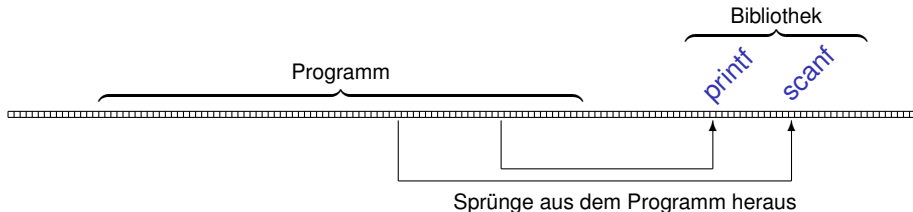
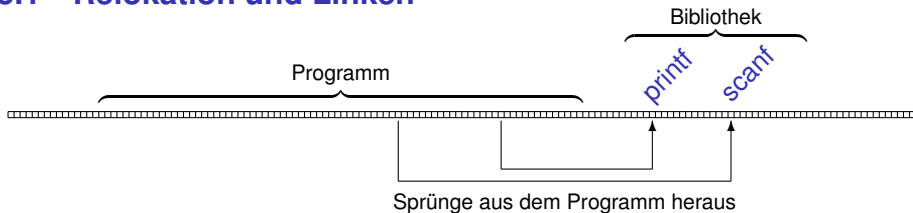


5.1 Relokation und Linken



Sprünge anpassen: Linken

5.1 Relokation und Linken



Sprünge anpassen: Linken

Beim Erzeugen der Datei: statisches Linken

Beim Laden: dynamisches Linken

Man kann Maschinenprogramme nicht „einfach so“ in den Speicher laden.

Man kann Maschinenprogramme nicht „einfach so“ in den Speicher laden.

Sprünge anpassen

- Relokation: Relokationstabelle
- Linken: Symboltabelle

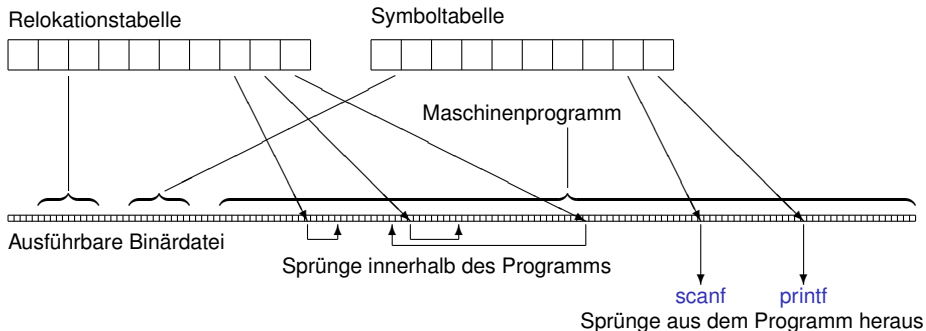
5.2 Dateiformate

Man kann Maschinenprogramme nicht „einfach so“ in den Speicher laden.

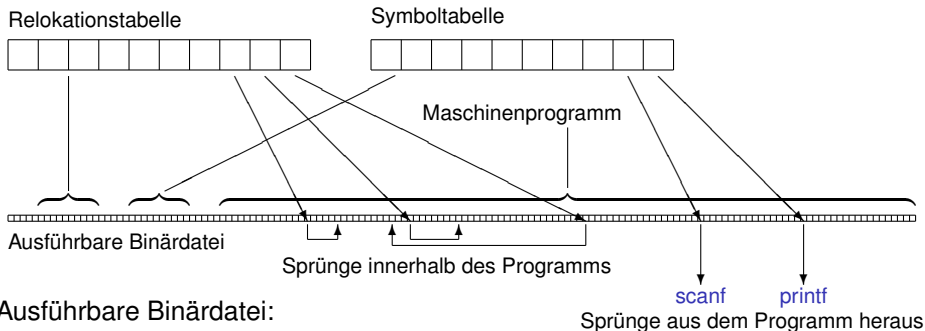
Sprünge anpassen

- Relokation: Relokationstabelle
- Linken: Symboltabelle

5.2 Dateiformate



5.2 Dateiformate



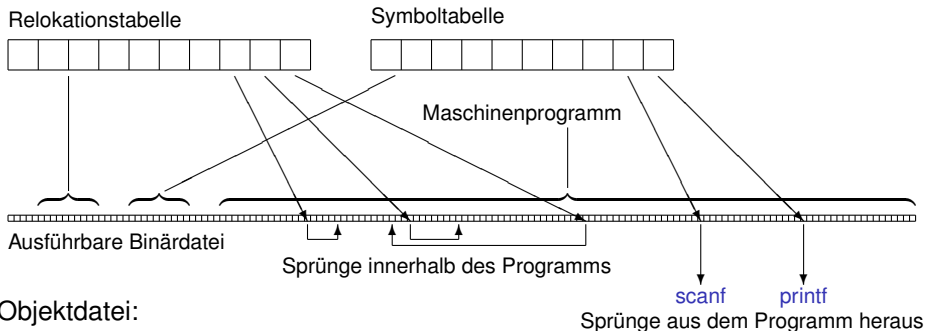
Ausführbare Binärdatei:

Relokationstabelle,
Symboltabelle für dynamischen Linker

Formate: a.out, COFF, ELF, ...

Dateiendungen: (keine), .elf, .com, .exe, .scr

5.2 Dateiformate



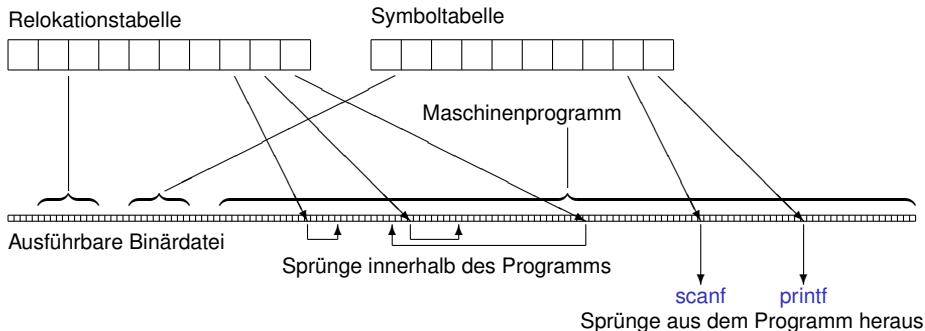
Objektdatei:

Relokationstabelle,
Symboltabellen für statischen und dynamischen Linker

Formate: a.out, COFF, ELF, ...

Dateiendungen: .o, .obj

5.2 Dateiformate



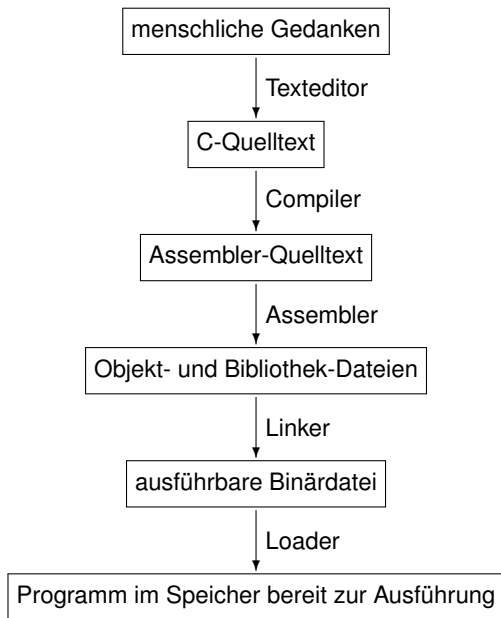
Bibliothek:

Zusammenfassung mehrerer Objekt-Dateien

Statische Bibliotheken: .a, .lib

Dynamische Bibliotheken: .so, .dll

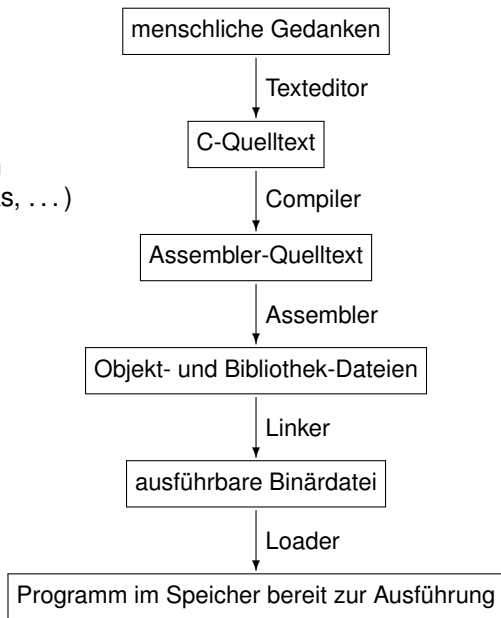
5.3 Die Toolchain



5.3 Die Toolchain

Automatischer Aufruf:

- Entwicklungsumgebungen
(z. B. Eclipse, Code::Blocks, ...)
- `gcc` = Compiler
+ Assembler
+ Linker
+ ...
- `make` kann *alles* aufrufen



5.4 Besonderheiten von Mikro-Controllern

Kein Betriebssystem

5.4 Besonderheiten von Mikro-Controllern

Kein Betriebssystem

→ kein Relocator, kein dynamischer Linker

5.4 Besonderheiten von Mikro-Controllern

Kein Betriebssystem

—→ kein Relocator, kein dynamischer Linker

—→ Wir müssen dem Mikro-Controller alles „mundgerecht“ servieren.

5.4 Besonderheiten von Mikro-Controllern

Kein Betriebssystem

→ kein Relocator, kein dynamischer Linker

→ Wir müssen dem Mikro-Controller alles „mundgerecht“ servieren.

- fertiges ROM: Hersteller
- Flash-Speicher und In-System Programmer (ISP)
- Flash-Speicher und Boot-Loader

In jedem Fall: statisch linken, Relokation vorher

→ ELF-Datei in HEX-Datei umwandeln

Format: Intel-Hex-Format

Dateiendung: .hex