

Grundlagen Rechnertechnik

Prof. Dr. Peter Gerwinski

10. Januar 2013

Grundlagen Rechnertechnik

- 1 Einführung**
- 2 Vom Schaltkreis zum Computer**
- 3 Architekturmerkmale von Prozessoren**
- 4 Der CPU-Stack**
- 5 Hardwarenahe Programmierung**
 - 5.1** Bit-Operationen
 - 5.2** I/O-Ports
 - 5.3** Interrupts
 - 5.4** volatile-Variable
 - 5.5** Software-Interrupts
 - 5.6** Byte-Reihenfolge – Endianness
 - 5.7** Speicherausrichtung – Alignment
- 6 Anwender-Software**
- 7 Bus-Systeme**
- 8 Pipelining**
- 9 Ausblick**

5.6 Byte-Reihenfolge – Endianness

5.6.1 Konzept

Eine Zahl geht über mehrere Speicherzellen.

Beispiel: 16-Bit-Zahl in 2 8-Bit-Speicherzellen

Welche Bits liegen wo?

$$1027 = 1024 + 2 + 1 = 0000\ 0100\ 0000\ 0011_2 = 0403_{16}$$

Speicherzellen:

04	03
----	----

 Big-Endian „großes Ende zuerst“
für Menschen leichter lesbar

03	04
----	----

 Little-Endian „kleines Ende zuerst“
bei Additionen effizienter

→ Geschmackssache

... **außer bei Datenaustausch!**

5.6 Byte-Reihenfolge – Endianness

5.6.1 Konzept

Eine Zahl geht über mehrere Speicherzellen.

Beispiel: 16-Bit-Zahl in 2 8-Bit-Speicherzellen

Welche Bits liegen wo?

—→ Geschmackssache

... **außer bei Datenaustausch!**

- Dateiformate
- Datenübertragung

5.6 Byte-Reihenfolge – Endianness

5.6.2 Dateiformate

Audio-Formate: Reihenfolge der Bytes in 16- und 32-Bit-Zahlen

- RIFF-WAVE-Dateien ([.wav](#)): Little-Endian
- Au-Dateien ([.au](#)): Big-Endian
- ältere AIFF-Dateien ([.aiff](#)): Big-Endian
- neuere AIFF-Dateien ([.aiff](#)): Little-Endian

Grafik-Formate: Reihenfolge der Bits in den Bytes

- PBM-Dateien: Big-Endian
- XBM-Dateien: Little-Endian

5.6 Byte-Reihenfolge – Endianness

5.6.2 Dateiformate

Audio-Formate: Reihenfolge der Bytes in 16- und 32-Bit-Zahlen

- RIFF-WAVE-Dateien (.wav): Little-Endian
- Au-Dateien (.au): Big-Endian
- ältere AIFF-Dateien (.aiff): Big-Endian
- neuere AIFF-Dateien (.aiff): Little-Endian

Grafik-Formate: Reihenfolge der Bits in den Bytes

- PBM-Dateien: Big-Endian, MSB first
- XBM-Dateien: Little-Endian, LSB first

MSB/LSB = most/least significant bit

5.6 Byte-Reihenfolge – Endianness

5.6.3 Datenübertragung

- RS-232 (serielle Schnittstelle): MSB first
- I²C: LSB first
- USB: beides
- Ethernet: LSB first
- TCP/IP (Internet): Big-Endian

5.7 Speicherausrichtung – Alignment

```
#include <stdint.h>
```

```
uint8_t a;  
uint16_t b;  
uint8_t c;
```

Speicheradresse durch 2 teilbar – „16-Bit-Alignment“

- 2-Byte-Operation: effizienter
- ... oder sogar nur dann erlaubt

→ Compiler optimiert Speicherausrichtung

```
uint8_t a;  
uint8_t dummy;  
uint16_t b;  
uint8_t c;  
  
uint8_t a;  
uint8_t c;  
uint16_t b;
```


5.7 Speicherausrichtung – Alignment

```
#include <stdint.h>
```

```
uint8_t a;  
uint16_t b;  
uint8_t c;
```

Speicheradresse durch 2 teilbar – „16-Bit-Alignment“

- 2-Byte-Operation: effizienter
- ... oder sogar nur dann erlaubt

→ Compiler optimiert Speicherausrichtung

```
uint8_t a;  
uint8_t dummy;  
uint16_t b;  
uint8_t c;  
  
uint8_t a;  
uint8_t c;  
uint16_t b;
```

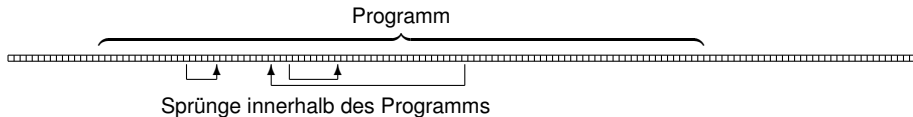
Fazit:

- Adressen von Variablen sind systemabhängig
- Bei Definition von Datenformaten Alignment beachten → effizienter

6 Anwender-Software

6.1 Relokation und Linken

Software im Speicher



6 Anwender-Software

6.1 Relokation und Linken

Software im Speicher

