Lucerne University of
Applied Sciences and Arts

HOCHSCHULE
LUZERN

Computer Science and
Information Technology

OSSEC-Labor: Einleitung Buffer Overflow

FH Zentralschweiz

Inhalt Heute

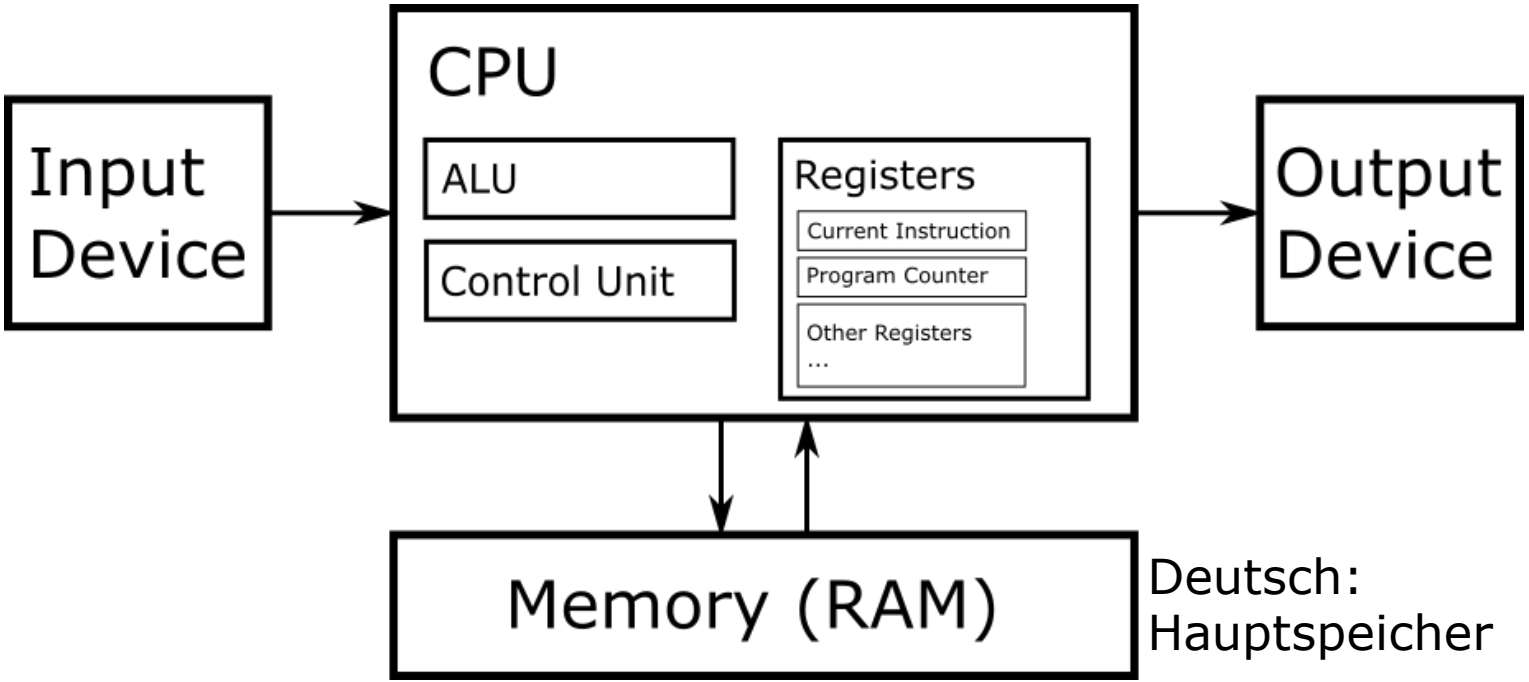
- Hintergrundwissen in Computer-Architektur für Labs

- Basics, bekannt aus REVE1-Blockkurs oder IRFORENSIC im Frühling

- Damit alle auf dem gleichen Stand sind

COMPUTER-ARCHITEKTUR

Von-Neumann-Maschine: Schematischer Aufbau eines Computers



Deutsch:
Hauptspeicher

Von-Neumann-Maschine: Funktionsweise der CPU

1. Instruktion aus dem Hauptspeicher laden
2. Instruction Pointer (Program Counter) erhöhen
3. Instruktion ausführen (Register ändern sich)
   - Wert aus Hauptspeicher in Register laden
   - Wert aus Register in Hauptspeicher schreiben
   - Mit Werten in Registern rechnen oder vergleichen
   - Programmfluss ändern (Jump, Verzweigung, Call)
4. Beginne von vorn.

Von-Neumann-Maschine: Hauptspeicher

Im Hauptspeicher (RAM) befinden sich:

- Daten
- Instruktionen

Welche Daten und Instruktionen sind wo im RAM?

- Speicher ist adressiert
- Im Programm: Zeiger (Pointer) auf Adressen im Speicher
- In der CPU: Adresse steht in einem Register

INSTRUKTIONEN IM SPEICHER

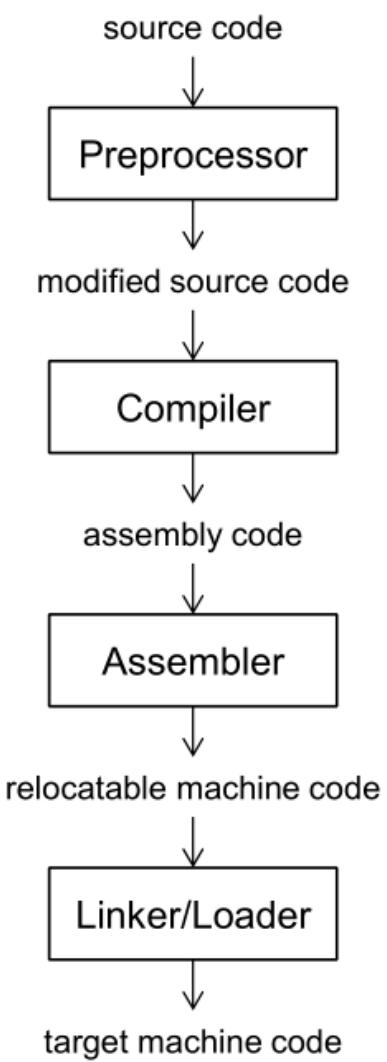
Binaries = Executables

- Ausführbares Programm
- Enthält Instruktionen und Daten
- Instruktionssatz durch CPU-Architektur vorgegeben
- Anordnung innerhalb des Programms betriebssystemabhängig

Von Code zum ausführbaren Programm

- Code wird in einer Programmiersprache wie C geschrieben
- Von Compiler zu Assembly-Code übersetzt
- Assembler macht daraus Maschinen-Code

Präprozessor und Linker sind momentan für uns nicht interessant



Von Code zum ausführbaren Programm: Compiler

C-Code

```
int square(int num)
{
    return num * num;
}
```

Assembly

```
square:
        pushq   %rbp
        movq    %rsp, %rbp
        movl    %edi, -4(%rbp)
        movl    -4(%rbp), %eax
        imull   %eax, %eax
        popq    %rbp
        ret
```

Von Code zum ausführbaren Programm: Assembler

Assembly

```
square:
        pushq   %rbp
        movq    %rsp, %rbp
        movl    %edi, -4(%rbp)
        movl    -4(%rbp), %eax
        imull   %eax, %eax
        popq    %rbp
        ret
```

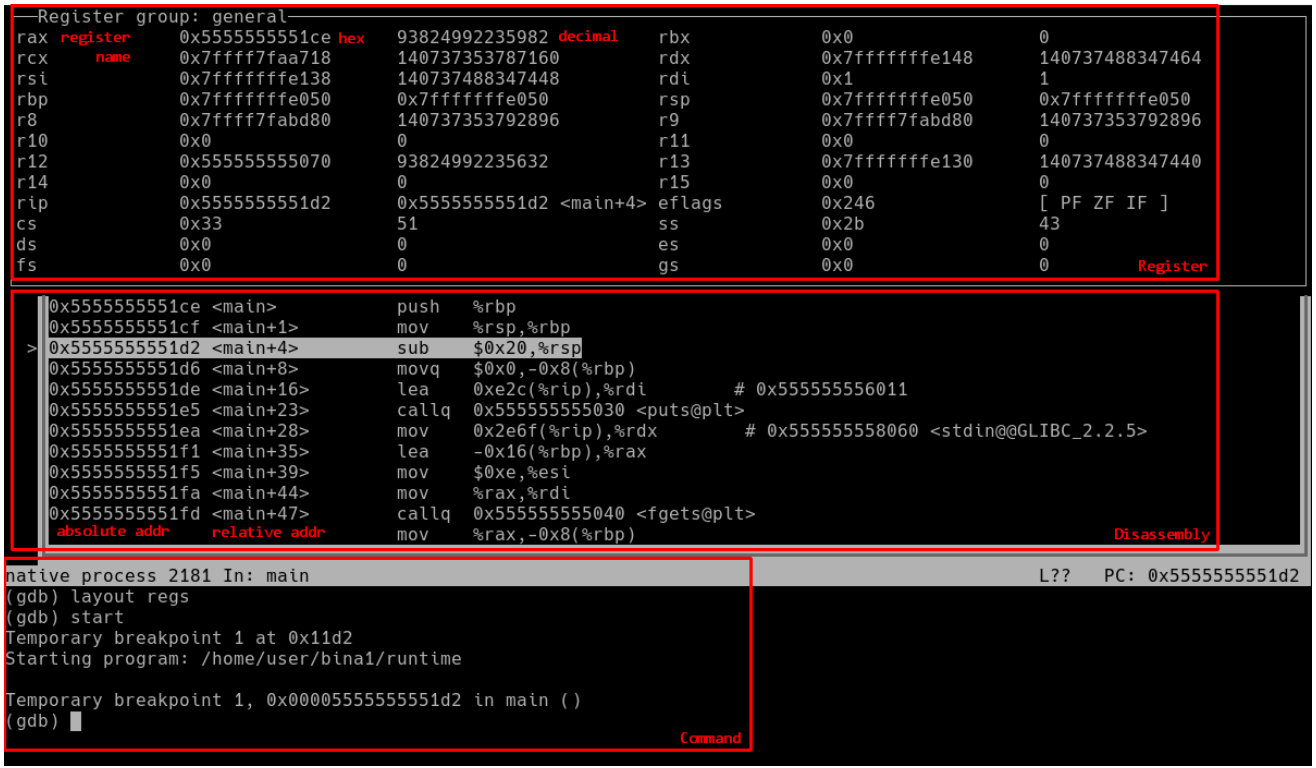
Maschinen-code

```
01010101 01001000 10001001 11100101
10001001 01111101 11111100 10001011
01000101 11111100 00001111 10101111
11000000 01011101 11000011
```

In Hex:

```
55 48 89 e5
89 7d fc 8b
45 fc 0f af
c0 5d c3
```

Laufendes Programm (Ansicht im Debugger)



CPU: Ansicht der Register

Memory: Ansicht des Programms im Speicher

Debugger Interface
(ist uns momentan egal)

DATEN IM SPEICHER

Hexadezimalwerte (Hex-Werte)

- Zahlen zur Basis 16
- Ziffern von 0 bis f: 0-9, a, b, c, d, e, f
- Grossbuchstaben oder Kleinbuchstaben
- Zur Klarifikation oft mit Präfix «0x» geschrieben
- Mit genau zwei Stellen können genau 256 Werte dargestellt werden: 0x00 bis 0xff
- $256 = 2^8$ = Anzahl Werte in einem Byte (8 bit)
- Beispiel: 0x2004 (Hex) = 8196 (Dezimal)
- 0x2004 hat vier Stellen, der Wert füllt also zwei Bytes (16 Bit)

Bytes und Adressen

- Ein Byte ist die kleinste *adressierbare* Speichereinheit
- Eine Adresse zeigt also immer auf ein Byte im Speicher

Werte in mehreren Bytes

- 1 Byte (8 Bit): «char», kleinste *adressierbare* Speichereinheit
- 2 Bytes: «short», «single word», 16-bit-Wort
- 4 Bytes: «long», «double word», 32-bit-Wort
- 8 Bytes: «long long», «quad word», 64-bit-Wort

Wortbreite

- Moderne CPUs haben eine 64-bit-Architektur
- Bedeutet: Register sind 64 Bit, also 8 Byte, gross
- Die CPU rechnet also grundsätzlich mit 64-bit-Werten
- 8 Bytes: «long long», «quad word», 64-bit-Wort