# Technische Informatik

## Ein C++ Programm aus sicht der CPU

Friedrich

12. Oktober 2017

**Programmcode / Algorithmus** Ein in einer (einigermaßen) verständlichen Programmiersprache geschriebene Algorithmus / Programm.

**Assembler / Maschienen Befehle** Das Programm runter gebrochen auf die vom Computer verstandenen Grund Operationen.

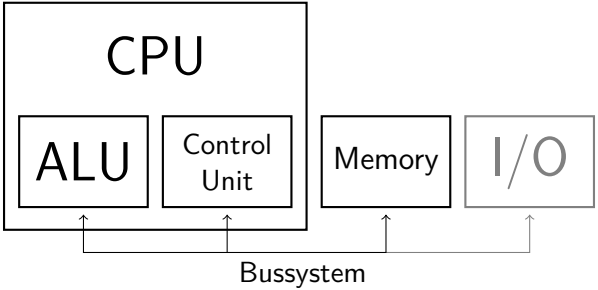
**Bitweise Operationen** Aufbau der Rechen Operationen ($+$, $-$, $\times$, $\ldots$) und anderen Befehlen die in Assembler verwendet werden aus den Logik Gattern (`and`, `or`, `xor`, …).

**Physikalisch** Physikalischer aufbau der Logik Gatter aus Transistoren

# Ein einfaches C++ Programm

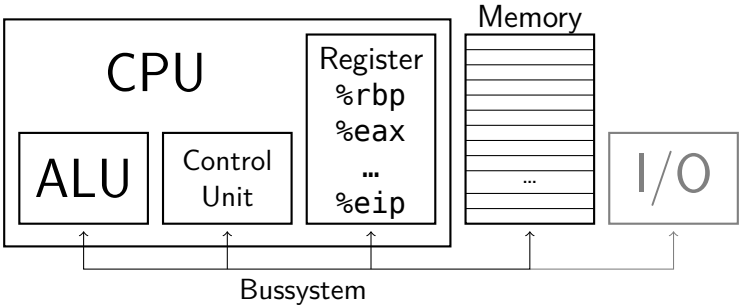
```
1  int main() {
2          int a = 7;
3          int b = 6;
4          a = a + b;
5          return 0;
6  }
```

# Van Neuman Architektur



# Van Neuman Architektur

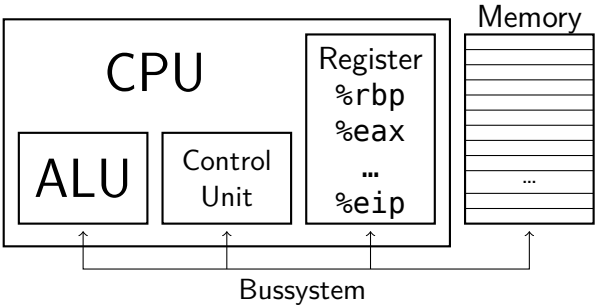
## mit für uns relevanten Teilen



# Der Programmablauf

## auf Ebene der Register und des Speichers

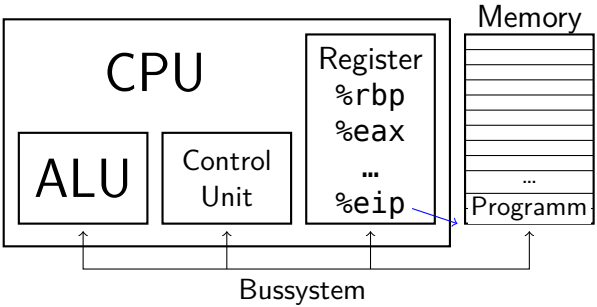
```
1  int main() {
2          int a = 7;
3          int b = 6;
4          a = a + b;
5          return 0;
6  }
```



# Der Programmablauf

## auf Ebene der Register und des Speichers

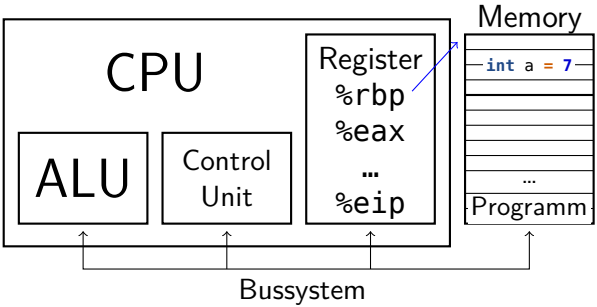
```
1  int main() {
2          int a = 7;
3          int b = 6;
4          a = a + b;
5          return 0;
6  }
```



# Der Programmablauf

## auf Ebene der Register und des Speichers

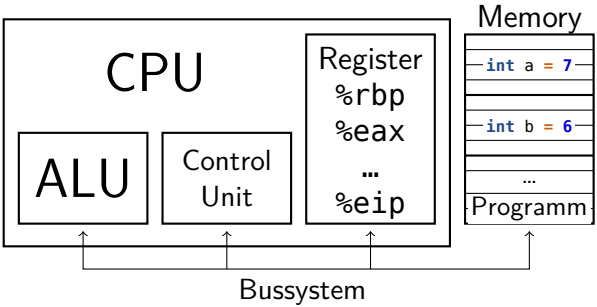
```
1  int main() {
2          int a = 7;
3          int b = 6;
4          a = a + b;
5          return 0;
6  }
```



# Der Programmablauf

## auf Ebene der Register und des Speichers

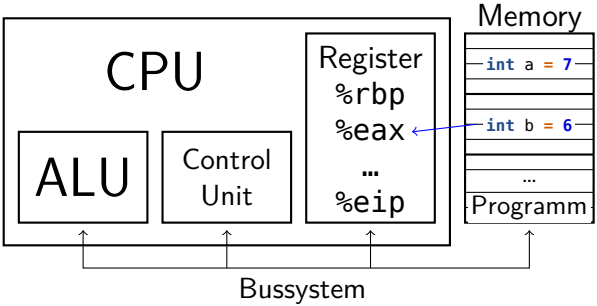
```
1  int main() {
2          int a = 7;
3          int b = 6;
4          a = a + b;
5          return 0;
6  }
```



# Der Programmablauf

## auf Ebene der Register und des Speichers

```
1  int main() {
2          int a = 7;
3          int b = 6;
4          a = a + b;
5          return 0;
6  }
```

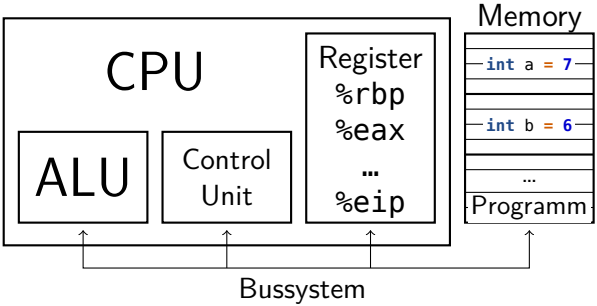



$$\texttt{b} \mapsto \texttt{\%eax} \quad \longrightarrow \quad \texttt{\%eax} + \texttt{a} \mapsto \texttt{a}$$

# Der Programmablauf

## auf Ebene der Register und des Speichers

```
1  int main() {
2          int a = 7;
3          int b = 6;
4          a = a + b;
5          return 0;
6  }
```

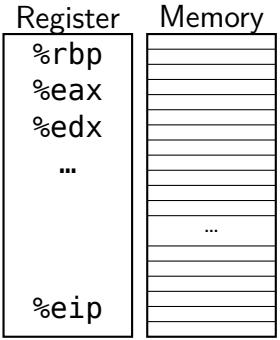



%eax = **0**

# Eine Variation des Programms

## Eine Aufgabe zum selbst Bearbeiten

```
1  int main() {
2          int a = 7;
3          int b = 6;
4          int c = a + b;
5          return c;
6  }
```



# Binäre Addition

# Binäre Zahlendarstellung

`int a` $= 0111_2 = (0 \times 2^3 + 1 \times 2^2 + 1 \times 2^1 + 1 \times 2^0)_{10} = 7_{10}$

`int b` $= 0110_2 = (0 \times 2^3 + 1 \times 2^2 + 1 \times 2^1 + 0 \times 2^0)_{10} = 6_{10}$

# Binäre Addition

$$
\begin{array}{cccc}
 & 0 & 1 & 1 & 1 \\
+ & 0 & 1 & 1 & 0 \\
\hline
\end{array}
$$

# Binäre Addition

$$
\begin{array}{cccc}
 & 0 & 1 & 1 & 1 \\
+ & 0_1 & 1_1 & 1 & 0 \\
\hline
 & 1 & 1 & 0 & 1
\end{array}
$$

# Binäre Addition

$$
\begin{array}{cccc}
 & 0 & 1 & 1 & 1 \\
+ & 0_1 & 1_1 & 1 & 0 \\
\hline
 & 1 & 1 & 0 & 1
\end{array}
$$

Logik Tafel (1 Bit)

| a | b | S | C |
|---|---|---|---|
| 0 | 0 | 0 | 0 |
| 0 | 1 | 1 | 0 |
| 1 | 0 | 1 | 0 |
| 1 | 1 | 0 | 1 |

# Binäre Addition

$$
\begin{array}{cccc}
 & 0 & 1 & 1 & 1 \\
+ & 0_1 & 1_1 & 1 & 0 \\
\hline
 & 1 & 1 & 0 & 1
\end{array}
$$

Ohne Übertrag

$$
\begin{array}{cccc}
 & 0 & 1 & 1 & 1 \\
+ & 0_1 & 1_1 & 1_0 & 0_0 \\
\hline
 & 0 & 0 & 0 & 1
\end{array}
$$

Logik Tafel (1 Bit)

| a | b | S | C |
|---|---|---|---|
| 0 | 0 | 0 | 0 |
| 0 | 1 | 1 | 0 |
| 1 | 0 | 1 | 0 |
| 1 | 1 | 0 | 1 |

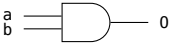
Nennt sich Half Adder

# Logik Gatter

## OR


| a | b | 0 |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 1 |

## AND

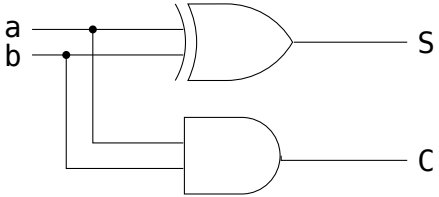



| a | b | 0 |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 1 | 0 |
| 1 | 0 | 0 |
| 1 | 1 | 1 |

## XOR


| a | b | 0 |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 0 |

# Half Adder




S Das Ergebnis von $a + b$ für ein Bit

C Ein Überlaufbit falls $a + b > 1$

# Full Adder

## zum selbst Basteln

| a | b | $C_{in}$ | S | $C_{out}$ |
|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 |
| 0 | 1 | 0 | 1 | 0 |
| 1 | 0 | 0 | 1 | 0 |
| 1 | 1 | 0 | 0 | 1 |
| 0 | 0 | 1 | 1 | 0 |
| 0 | 1 | 1 | 0 | 1 |
| 1 | 0 | 1 | 0 | 1 |
| 1 | 1 | 1 | 1 | 1 |

# Full Adder

## zum selbst Basteln

| a | b | $C_{in}$ | S | $C_{out}$ |
|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 |
| 0 | 1 | 0 | 1 | 0 |
| 1 | 0 | 0 | 1 | 0 |
| 1 | 1 | 0 | 0 | 1 |
| 0 | 0 | 1 | 1 | 0 |
| 0 | 1 | 1 | 0 | 1 |
| 1 | 0 | 1 | 0 | 1 |
| 1 | 1 | 1 | 1 | 1 |

### OR


| a | b | 0 |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 1 |

### AND


| a | b | 0 |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 1 | 0 |
| 1 | 0 | 0 |
| 1 | 1 | 1 |

### XOR


| a | b | 0 |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 0 |