

# x86-Assembly

amd64 unter Linux

[github.com/khde](https://github.com/khde)

11. November 2023



# Inhaltsverzeichnis

<b>I</b>	<b>Grundlagen</b>	<b>9</b>
<b>II</b>	<b>Einstieg in Assembly</b>	<b>11</b>
<b>1</b>	<b>Aufbau</b>	<b>13</b>
1.1	Struktur . . . . .	13
1.1.1	.text Sektion . . . . .	13
1.1.2	.data Sektion . . . . .	13
1.1.3	.bss Sektion . . . . .	13
1.1.4	Weitere Sektionen . . . . .	13
1.2	Syntax . . . . .	13
1.3	Adressiermoduse . . . . .	13
1.4	Erstes Programm . . . . .	13
<b>2</b>	<b>Datenübertragung</b>	<b>15</b>
2.1	mov . . . . .	15
2.2	lea . . . . .	15
2.3	xchg . . . . .	15
<b>3</b>	<b>Ganzzahlige Arithmetik</b>	<b>17</b>
3.1	Addieren . . . . .	18
3.1.1	add . . . . .	18
3.1.2	adc . . . . .	18
3.1.3	xadd . . . . .	18
3.2	Subtrahieren . . . . .	18
3.2.1	sub . . . . .	18
3.2.2	sbb . . . . .	18
3.3	Multiplizieren . . . . .	18
3.3.1	mul . . . . .	18
3.3.2	imul . . . . .	18
3.4	Dividieren . . . . .	18

3.4.1	div . . . . .	18
3.4.2	idiv . . . . .	18
3.5	Inkrement inc . . . . .	18
3.6	Dekrement dec . . . . .	18
<b>4</b>	<b>Logik</b>	<b>19</b>
4.1	and . . . . .	19
4.2	or . . . . .	19
4.3	not . . . . .	19
4.4	xor . . . . .	19
4.5	test . . . . .	19
<b>5</b>	<b>Bedingtes Ausführen</b>	<b>21</b>
5.1	Springen jmp . . . . .	21
5.2	Bedingtes Springen cmp, jCC . . . . .	21
5.3	rflags-Register . . . . .	21
5.4	loop, loopne . . . . .	21
5.5	rflags Manipulieren . . . . .	21
5.5.1	Carry-Flag . . . . .	21
5.5.2	Direction-Flag . . . . .	21
5.5.3	Interrupt-Flag . . . . .	21
5.6	lahf, sahf . . . . .	21
<b>6</b>	<b>Bit Operationen</b>	<b>23</b>
6.1	Bit Shifting . . . . .	23
6.1.1	Logischer Shift: shr, shl . . . . .	23
6.1.2	Arithmetischer Shift: sar, sal . . . . .	23
6.1.3	shld, shrd . . . . .	23
6.2	Bit Rotation . . . . .	23
6.2.1	ror . . . . .	23
6.2.2	rol . . . . .	23
6.2.3	rcr . . . . .	23
6.2.4	rcl . . . . .	23
<b>7</b>	<b>Bit Manipulation</b>	<b>25</b>
7.1	bts . . . . .	25
7.2	btr . . . . .	25
7.3	bt . . . . .	25
7.4	bswap . . . . .	25

<b>8</b>	<b>Stack</b>	<b>27</b>
8.1	Grundlagen des Stacks . . . . .	27
8.2	push, pop . . . . .	27
8.3	Die Rolle von rsp, rbp . . . . .	27
8.4	Stack Frame und Stack alignment . . . . .	27
<b>9</b>	<b>Funktionen</b>	<b>29</b>
9.1	Funktionen mit call und ret . . . . .	29
9.2	Funktionsprolog und Funktionsepilog . . . . .	29
9.3	Aufrufskonventionen . . . . .	29
9.4	Registerinhalt sichern . . . . .	29
<b>10</b>	<b>Systemaufrufe</b>	<b>31</b>
10.1	syscall, int 0x80 . . . . .	31
<b>11</b>	<b>Zeichenketten und Felder</b>	<b>33</b>
11.1	stosb/stosw/stosd . . . . .	33
11.2	movsb, rep stosb . . . . .	33
<b>12</b>	<b>Kommandozeilenargumente</b>	<b>35</b>
<b>13</b>	<b>Weitere Befehle</b>	<b>37</b>
13.1	nop . . . . .	37
13.2	ASCII und BCD . . . . .	37
13.2.1	aaa . . . . .	37
13.2.2	aas . . . . .	37
13.2.3	aad . . . . .	37
13.2.4	aam . . . . .	37
13.2.5	daa . . . . .	37
13.2.6	das . . . . .	37
<b>A</b>	<b>Wichtige Befehle</b>	<b>39</b>
<b>B</b>	<b>Wichtige Linux-Syscalls</b>	<b>41</b>
<b>C</b>	<b>Register</b>	<b>43</b>
<b>D</b>	<b>Erweiterungen</b>	<b>45</b>
<b>E</b>	<b>ASCII-Tabelle</b>	<b>47</b>



# Vorwort





# Teil I

## Grundlagen



# Teil II

## Einstieg in Assembly



# Kapitel 1

## Aufbau

### 1.1 Struktur

#### 1.1.1 .text Sektion

#### 1.1.2 .data Sektion

#### 1.1.3 .bss Sektion

#### 1.1.4 Weitere Sektionen

### 1.2 Syntax

Intel vs AT&T Opcode, Mnemonic

### 1.3 Adressiermoduse

Register, Speicher, Direkt

### 1.4 Erstes Programm

Hallo Welt



# Kapitel 2

## Datenübertragung

2.1 mov

2.2 lea

2.3 xchg







## Kapitel 3

# Ganzzahlige Arithmetik

### 3.1 Addieren

#### 3.1.1 add

#### 3.1.2 adc

#### 3.1.3 xadd

### 3.2 Subtrahieren

#### 3.2.1 sub

#### 3.2.2 sbb

### 3.3 Multiplizieren

#### 3.3.1 mul

#### 3.3.2 imul

### 3.4 Dividieren

#### 3.4.1 div

#### 3.4.2 idiv

### 3.5 Inkrement inc

### 3.6 Dekrement dec

# Kapitel 4

## Logik

4.1 and

4.2 or

4.3 not

4.4 xor

4.5 test



# Kapitel 5

## Bedingtes Ausführen

### 5.1 Springen jmp

### 5.2 Bedingtes Springen cmp, jCC

### 5.3 rflags-Register

jz, jnz, je, jne, jg, jge, jl, jle, ja, jae, jb, jbe, jo, js, jecxz

### 5.4 loop, loopne

### 5.5 rflags Manipulieren

#### 5.5.1 Carry-Flag

stc, clc, cmc

#### 5.5.2 Direction-Flag

std, cld

#### 5.5.3 Interrupt-Flag

sti, cli

### 5.6 lahf, sahf



# Kapitel 6

## Bit Operationen

### 6.1 Bit Shifting

6.1.1 Logischer Shift: shr, shl

6.1.2 Arithmetischer Shift: sar, sal

6.1.3 shld, shrd

### 6.2 Bit Rotation

6.2.1 ror

6.2.2 rol

6.2.3 rcr

6.2.4 rcl





# Kapitel 7

## Bit Manipulation

7.1    `bts`

7.2    `btr`

7.3    `bt`

7.4    `bswap`

`setCC`



# Kapitel 8

## Stack

### 8.1 Grundlagen des Stacks

### 8.2 push, pop

### 8.3 Die Rolle von rsp, rbp

### 8.4 Stack Frame und Stack alignment

```
push rbp  
mov rbp, rsp
```



# Kapitel 9

## Funktionen

### 9.1 Funktionen mit call und ret

### 9.2 Funktionsprolog und Funktionsepilog

enter, leave

### 9.3 Aufrufskonventionen

System V AMD64 ABI

### 9.4 Registerinhalt sichern

-caller, callee saved

-push eax

equivalent zu

sub esp, 4

mov [esp], eax

-pop eax

sub esp, 4

mov [esp], eax



# Kapitel 10

## Systemaufrufe

-Kernel im Ring 0 (Kernel-Modus) -Zugriff auf gesamten CPU-Befehlssatz und Speicherbereich -Normaler Prozess läuft im unprivilegierten Ringen 1 - 3 (Benutzer-Modus)

### 10.1 syscall, int 0x80





# Kapitel 11

## Zeichenketten und Felder

11.1 stosb/stosw/stosd

11.2 movsb, rep stosb



# Kapitel 12

## Kommandozeilenargumente

```
int main(int argc, char *argv[]) rdi rsi
```



# Kapitel 13

## Weitere Befehle

### 13.1 nop

`nop xchg rax, rax mov rax, rax`

### 13.2 ASCII und BCD

#### 13.2.1 aaa

#### 13.2.2 aas

#### 13.2.3 aad

#### 13.2.4 aam

#### 13.2.5 daa

#### 13.2.6 das

-Nicht Vorhanden in 64-Bit Modus



## Anhang A

### Wichtige Befehle





## Anhang B

### Wichtige Linux-Syscalls



Anhang C

Register



# Anhang D

## Erweiterungen



# Anhang E

## ASCII-Tabelle