Tutorübung Grundlagen: Betriebssysteme und Systemsoftware

Moritz Beckel

München, 18. Januar 2023

Mittwoch 14:15-16:00 Uhr Online (https://bbb.in.tum.de/mor-6ij-iuw-ypm)

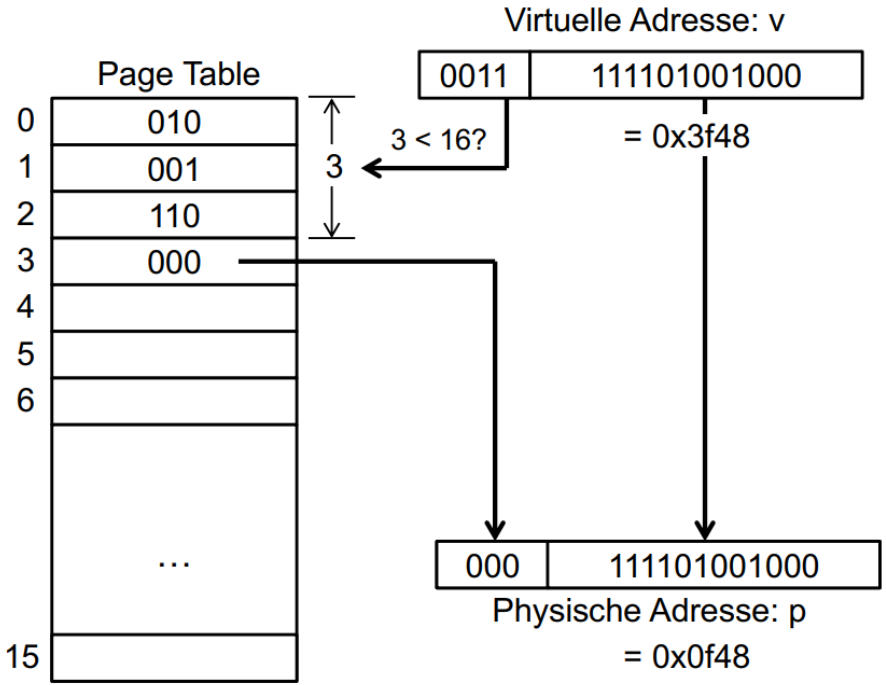
Zulip-Stream https://zulip.in.tum.de/#narrow/stream/1296-GBS-Mi-1400-A

Unterrichtsmaterialien findest du hier:

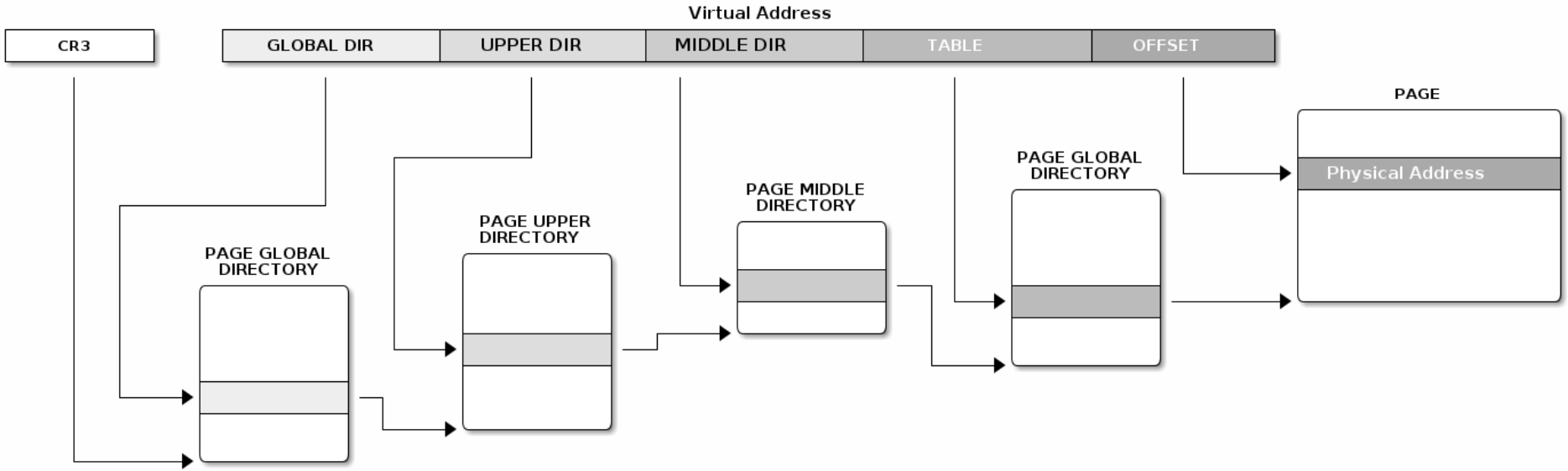
https://home.in.tum.de/~beckel/gbs

Folien wurden von mir selbst erstellt. Es besteht keine Garantie auf Korrektheit.

Paging – Adressabbildung

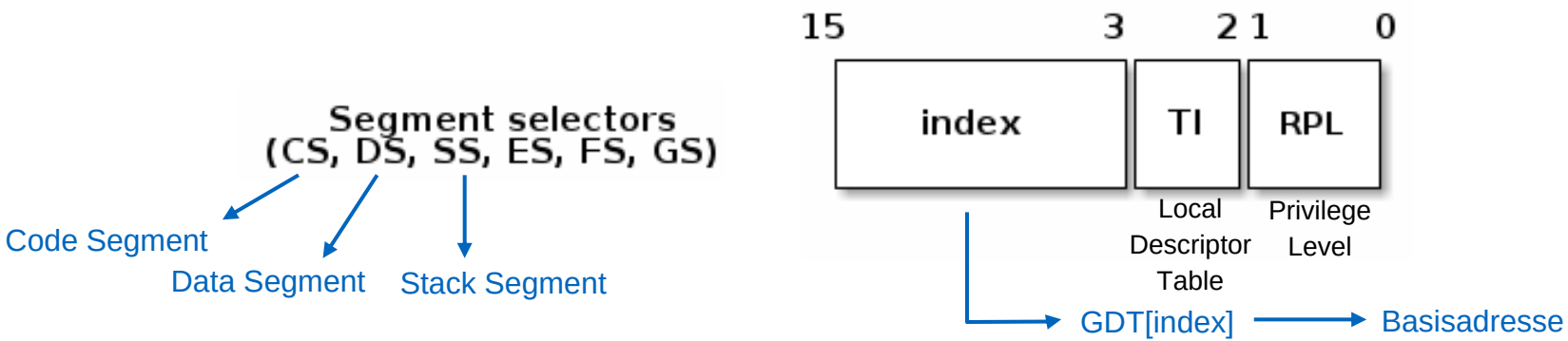


Paging – Mehrstufig

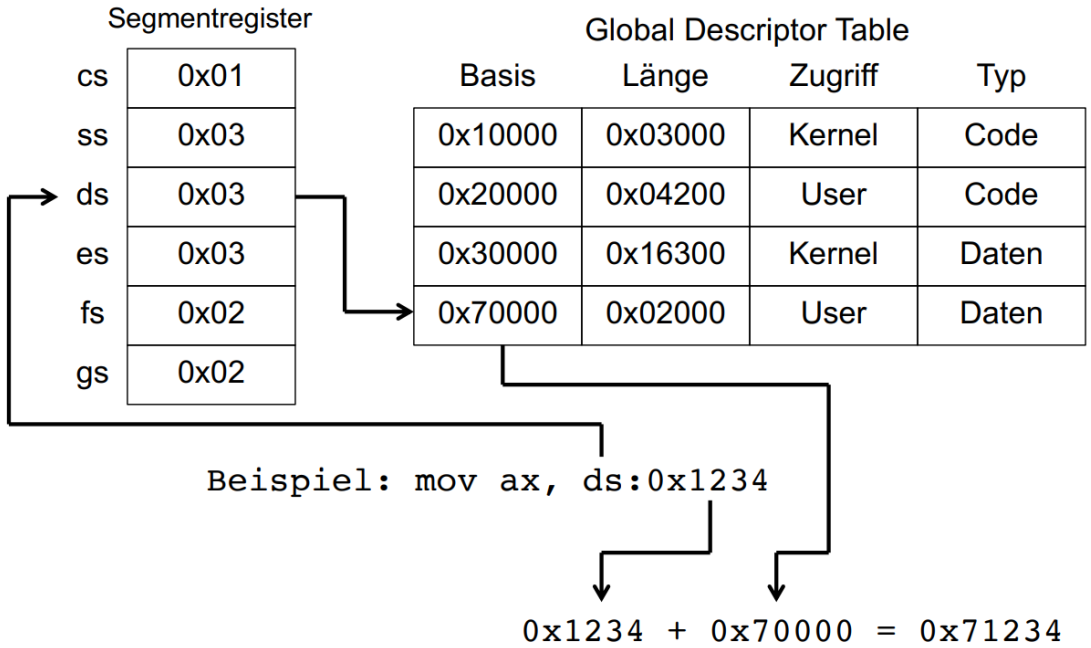


Segmentierung

1. Aufteilung des Adressraums in unterschiedliche Segmente (Code, Data, Stack, …)
2. Basisadressierung (logische Adresse + Basisadresse) der Segmente
3. Segmente werden über Segmentregister (Segment selectors) adressiert
4. Segmentregister zeigen auf die GDT (Global Descriptor Table), diese enthält Basisadresse



Segmentierung



Aufgabe 2

Wir betrachten ein vereinfachtes Beispiel zur Adressübersetzung: ein Computer mit 32-bit breiten virtuellen Adressen benutzt eine zweistufige Seitentabelle zur Adressübersetzung. Eine virtuelle Adresse bestehe aus 9 Bits für die erste Stufe der Adressübersetzung, 11 Bits für die zweite Stufe sowie einem Offset.

a) Skizzieren Sie den Aufbau einer virtuellen Adresse in diesem Szenario. Verdeutlichen Sie sich, wie die Adressübersetzung mit einer zweistufigen Seitentabelle mittels der einzelnen Bestandteile der Adresse durchgeführt wird.

Wir betrachten ein vereinfachtes Beispiel zur Adressübersetzung: ein Computer mit 32-bit breiten virtuellen Adressen benutzt eine zweistufige Seitentabelle zur Adressübersetzung. Eine virtuelle Adresse bestehe aus 9 Bits für die erste Stufe der Adressübersetzung, 11 Bits für die zweite Stufe sowie einem Offset.

Aufgabe 2

Wir betrachten ein vereinfachtes Beispiel zur Adressübersetzung: ein Computer mit 32-bit breiten virtuellen Adressen benutzt eine zweistufige Seitentabelle zur Adressübersetzung. Eine virtuelle Adresse bestehe aus 9 Bits für die erste Stufe der Adressübersetzung, 11 Bits für die zweite Stufe sowie einem Offset.

a) Skizzieren Sie den Aufbau einer virtuellen Adresse in diesem Szenario. Verdeutlichen Sie sich, wie die Adressübersetzung mit einer zweistufigen Seitentabelle mittels der einzelnen Bestandteile der Adresse durchgeführt wird.

Aufgabe 2

Wir betrachten ein vereinfachtes Beispiel zur Adressübersetzung: ein Computer mit 32-bit breiten virtuellen Adressen benutzt eine zweistufige Seitentabelle zur Adressübersetzung. Eine virtuelle Adresse bestehe aus 9 Bits für die erste Stufe der Adressübersetzung, 11 Bits für die zweite Stufe sowie einem Offset.

b) Wie groß sind die Seiten?

Aufgabe 2

Wir betrachten ein vereinfachtes Beispiel zur Adressübersetzung: ein Computer mit 32-bit breiten virtuellen Adressen benutzt eine zweistufige Seitentabelle zur Adressübersetzung. Eine virtuelle Adresse bestehe aus 9 Bits für die erste Stufe der Adressübersetzung, 11 Bits für die zweite Stufe sowie einem Offset.

b) Wie groß sind die Seiten?

- 32 – 9 – 11 = 12 Bit Offset
- 2^{12} B = 4096 B = 4 KiB Seitengröße

Aufgabe 2

Wir betrachten ein vereinfachtes Beispiel zur Adressübersetzung: ein Computer mit 32-bit breiten virtuellen Adressen benutzt eine zweistufige Seitentabelle zur Adressübersetzung. Eine virtuelle Adresse bestehe aus 9 Bits für die erste Stufe der Adressübersetzung, 11 Bits für die zweite Stufe sowie einem Offset.

c) Aus wie vielen Seiten besteht der virtuelle Adressraum?

Aufgabe 2

Wir betrachten ein vereinfachtes Beispiel zur Adressübersetzung: ein Computer mit 32-bit breiten virtuellen Adressen benutzt eine zweistufige Seitentabelle zur Adressübersetzung. Eine virtuelle Adresse bestehe aus 9 Bits für die erste Stufe der Adressübersetzung, 11 Bits für die zweite Stufe sowie einem Offset.

c) Aus wie vielen Seiten besteht der virtuelle Adressraum?

- 32 Bit – 12 Bit Offset = 20 Bit für Seitennummern
- 2^{20} Seiten

Aufgabe 2

Wir betrachten ein vereinfachtes Beispiel zur Adressübersetzung: ein Computer mit 32-bit breiten virtuellen Adressen benutzt eine zweistufige Seitentabelle zur Adressübersetzung. Eine virtuelle Adresse bestehe aus 9 Bits für die erste Stufe der Adressübersetzung, 11 Bits für die zweite Stufe sowie einem Offset.

d) Wie groß sind die Seitentabellen jeweils, wenn für die Größe eines Eintrags vereinfacht 8 Byte angenommen werden?

Aufgabe 2

Wir betrachten ein vereinfachtes Beispiel zur Adressübersetzung: ein Computer mit 32-bit breiten virtuellen Adressen benutzt eine zweistufige Seitentabelle zur Adressübersetzung. Eine virtuelle Adresse bestehe aus 9 Bits für die erste Stufe der Adressübersetzung, 11 Bits für die zweite Stufe sowie einem Offset.

d) Wie groß sind die Seitentabellen jeweils, wenn für die Größe eines Eintrags vereinfacht 8 Byte angenommen werden?

- Erste Stufe: 2^9 Einträge * 8 Bytes = 2^{12} B = 4 KiB
- Zweite Stufe: 2^9 Tabellen * 2^{11} Einträge * 8 Bytes = 2^{23} B = 8 MiB

Aufgabe 2

Wir betrachten ein vereinfachtes Beispiel zur Adressübersetzung: ein Computer mit 32-bit breiten virtuellen Adressen benutzt eine zweistufige Seitentabelle zur Adressübersetzung. Eine virtuelle Adresse bestehe aus 9 Bits für die erste Stufe der Adressübersetzung, 11 Bits für die zweite Stufe sowie einem Offset.

e) Nehmen wir an, wir verwenden statt der zweistufigen Übersetzung eine einstufige, bei der die erste Stufe 20-bit breite Seitennummern verwendet. Wie viel Speicher kann adressiert werden? Kann mittels der zweistufigen Übersetzung mehr Speicher adressiert werden?

Aufgabe 2

Wir betrachten ein vereinfachtes Beispiel zur Adressübersetzung: ein Computer mit 32-bit breiten virtuellen Adressen benutzt eine zweistufige Seitentabelle zur Adressübersetzung. Eine virtuelle Adresse bestehe aus 9 Bits für die erste Stufe der Adressübersetzung, 11 Bits für die zweite Stufe sowie einem Offset.

e) Nehmen wir an, wir verwenden statt der zweistufigen Übersetzung eine einstufige, bei der die erste Stufe 20-bit breite Seitennummern verwendet. Wie viel Speicher kann adressiert werden? Kann mittels der zweistufigen Übersetzung mehr Speicher adressiert werden?

- Einstufig: 2^{20} Seiten * 2^{12} Offset Adressen = 2^{32} Adressen
- Zweistufig: 2^{9} Seiten * 2^{11} Seiten * 2^{12} Offset Adressen = 2^{32} Adressen

Aufgabe 3

Wir wollen die Adressübersetzung beim Paging näher betrachten. Gegeben sei eine Hauptspeichergröße von 64 KiB. Es sollen 32 virtuelle Seiten adressiert werden. Die Zahl der Kacheln beträgt 8.

a) Wie lautet die höchste virtuelle Speicheradresse?

Seite	Kachel
0	0
1	1
2	2
3	3
4	-
5	-
6	-
7	-
8	-
9	7
10	-
11	4
12	5
13	6

Aufgabe 3

Wir wollen die Adressübersetzung beim Paging näher betrachten. Gegeben sei eine Hauptspeichergröße von 64 KiB. Es sollen 32 virtuelle Seiten adressiert werden. Die Zahl der Kacheln beträgt 8.

a) Wie lautet die höchste virtuelle Speicheradresse?

- Seiten-/Kachelgröße: 64 KiB / 8 Kacheln = 8 KiB pro Seite/Kachel
- Anzahl virtuelle Adressen: 32 * 8 Kib = 256 KiB
- Höchste Adresse: 256 KiB – 1 = 2^{18} B – 1 = 262.143 B

Seite	Kachel
0	0
1	1
2	2
3	3
4	-
5	-
6	-
7	-
8	-
9	7
10	-
11	4
12	5
13	6

Aufgabe 3

Wir wollen die Adressübersetzung beim Paging näher betrachten. Gegeben sei eine Hauptspeichergröße von 64 KiB. Es sollen 32 virtuelle Seiten adressiert werden. Die Zahl der Kacheln beträgt 8.

b) Wie viele Bit sind die virtuelle und physische Adresse jeweils breit?

Seite	Kachel
0	0
1	1
2	2
3	3
4	-
5	-
6	-
7	-
8	-
9	7
10	-
11	4
12	5
13	6

Aufgabe 3

Wir wollen die Adressübersetzung beim Paging näher betrachten. Gegeben sei eine Hauptspeichergröße von 64 KiB. Es sollen 32 virtuelle Seiten adressiert werden. Die Zahl der Kacheln beträgt 8.

b) Wie viele Bit sind für die virtuelle und physische Adresse jeweils breit?

- Bitanzahl Offset: $\log_2 8\text{KiB} = \log_2 2^{13}\,\text{B} = 13$ Bit
- Bitanzahl Seiten: $\log_2 32 = 5$
- Bitanzahl Kacheln: $\log_2 8 = 3$
- Bitanzahl virtuelle Adresse = 13 + 5 = 18 Bit
- Bitanzahl physische Adresse = 13 + 3 = 16 Bit

Seite	Kachel
0	0
1	1
2	2
3	3
4	-
5	-
6	-
7	-
8	-
9	7
10	-
11	4
12	5
13	6

Aufgabe 3

Wir wollen die Adressübersetzung beim Paging näher betrachten. Gegeben sei eine Hauptspeichergröße von 64 KiB. Es sollen 32 virtuelle Seiten adressiert werden. Die Zahl der Kacheln beträgt 8.
(13 Bit Offset, 5 Bit Seitennummer, 3 Bit Kachelnummer)

c) Es wird auf die folgenden virtuellen Adressen zugegriffen. Ermitteln Sie die jeweils angesprochene physische Adresse. Benutzen Sie die Pagetable rechts. Hinweise: Rechnen Sie im Folgenden mit der hexadezimal- und Binärdarstellung der Werte. Dies erleichtert die Unterteilung in Seiten-/Kachelnummer und Offset.

Zugriffe: 0x559, 0x1208c, 0x16001, 0x0a777, 0x13992

Seite	Kachel
0	0
1	1
2	2
3	3
4	-
5	-
6	-
7	-
8	-
9	7
10	-
11	4
12	5
13	6

Aufgabe 3

c) Es wird auf die folgenden virtuellen Adressen zugegriffen. Ermitteln Sie die jeweils angesprochene physische Adresse. Benutzen Sie die Pagetable rechts. Hinweise: Rechnen Sie im Folgenden mit der hexadezimal- und Binärdarstellung der Werte. Dies erleichtert die Unterteilung in Seiten-/Kachelnummer und Offset. (0x559, 0x1208c, 0x16001, 0x0a777, 0x13992)

Virtuelle Adresse	Seiten nummer	Offset	Kachel nummer	Physische Adresse

Seite	Kachel
0	0
1	1
2	2
3	3
4	-
5	-
6	-
7	-
8	-
9	7
10	-
11	4
12	5
13	6

Aufgabe 3

c) Es wird auf die folgenden virtuellen Adressen zugegriffen. Ermitteln Sie die jeweils angesprochene physische Adresse. Benutzen Sie die Pagetable rechts. Hinweise: Rechnen Sie im Folgenden mit der hexadezimal- und Binärdarstellung der Werte. Dies erleichtert die Unterteilung in Seiten-/Kachelnummer und Offset.

Virtuelle Adresse	Seiten nummer	Offset	Kachel nummer	Physische Adresse
0x00559 = 0000 0000 0101 0101 1001	0x0	0x0559	0x0	$0x0 * 0x2000 + 0x0559 = 0x559$
0x1208c = 0001 0010 0000 1000 1100	0x9	0x008c	0x7	$0x7 * 0x2000 + 0x008c = 0xe08c$
0x16001 = 0001 0110 0000 0000 0001	0xb	0x0001	0x4	$0x4 * 0x2000 + 0x0001 = 0x8001$
0x0a777 = 0000 1010 0111 0111 0111	0x5	0x0777	PF	Error
0x13992 = 0001 0011 1001 1001 0010	0x9	0x1992	0x7	$0x7 * 0x2000 + 0x1992 = 0xF992$

Seite	Kachel
0	0
1	1
2	2
3	3
4	-
5	-
6	-
7	-
8	-
9	7
10	-
11	4
12	5
13	6

Aufgabe 3

Wir wollen die Adressübersetzung beim Paging näher betrachten. Gegeben sei eine Hauptspeichergröße von 64 KiB. Es sollen 32 virtuelle Seiten adressiert werden. Die Zahl der Kacheln beträgt 8.
(13 Bit Offset, 5 Bit Seitennummer, 3 Bit Kachelnummer)

d) Welche virtuelle Adressen verwendet ein Programm, wenn Zugriffe auf die physischen Adressen erfolgen?

Zugriffe: 0x2000, 0x8235

Seite	Kachel
0	0
1	1
2	2
3	3
4	-
5	-
6	-
7	-
8	-
9	7
10	-
11	4
12	5
13	6

Aufgabe 3

d) Welche virtuelle Adressen verwendet ein Programm, wenn Zugriffe auf die physischen Adressen erfolgen? (0x2000, 0x8235)

Physische Adresse	Kachel nummer	Offset	Seiten nummer	Virtuelle Adresse

Seite	Kachel
0	0
1	1
2	2
3	3
4	-
5	-
6	-
7	-
8	-
9	7
10	-
11	4
12	5
13	6

Aufgabe 3

d) Welche virtuelle Adressen verwendet ein Programm, wenn Zugriffe auf die physischen Adressen erfolgen? (0x2000, 0x8235)

Physische Adresse	Kachel nummer	Offset	Seiten nummer	Virtuelle Adresse
0x2000 = 0010 0000 0000 0000	0x1	0x0	0x1	$0x1 * 0x2000 + 0 = 0x2000$
0x8235 = 1000 0010 0011 0101	0x4	0x235	0xb	$0xb * 0x2000 + 0x235 = 0x16235$

Seite	Kachel
0	0
1	1
2	2
3	3
4	-
5	-
6	-
7	-
8	-
9	7
10	-
11	4
12	5
13	6

Aufgabe 4

Wir betrachten im Folgenden vereinfacht Segmentadressierung unter x86. Jeder Eintrag der Global Descriptor Table ist dabei 8 Byte groß. Im Segmentregister wird dann der Offset des Eintrags gespeichert.

a) Erweitern Sie das Bild oben, sodass ein Zugriff auf gs:0x1000 auf einen Zugriff auf 0x40000 übersetzt wird.

Segmentregister

cs	0x08
ss	0x30
ds	0x28
es	0x10
fs	0x18
gs	

Global Descriptor Table

	Basis	Länge	Zugriff	Typ
0x0	0x10300	0x0e000	Kernel	Daten
0x8	0x10000	0x03000	Kernel	Code
0x10	0x20000	0x00800	Benutzer	Code
0x18	0x40000	0x13700	Kernel	Daten
0x20			Kernel	Daten
0x28	0x80000	0x22000	Benutzer	Daten

Aufgabe 4

Wir betrachten im Folgenden vereinfacht Segmentadressierung unter x86. Jeder Eintrag der Global Descriptor Table ist dabei 8 Byte groß. Im Segmentregister wird dann der Offset des Eintrags gespeichert.

a) Erweitern Sie das Bild oben, sodass ein Zugriff auf gs:0x1000 auf einen Zugriff auf 0x40000 übersetzt wird.

Segmentregister

cs	0x08
ss	0x30
ds	0x28
es	0x10
fs	0x18
gs	0x20

Global Descriptor Table

	Basis	Länge	Zugriff	Typ
0x0	0x10300	0x0e000	Kernel	Daten
0x8	0x10000	0x03000	Kernel	Code
0x10	0x20000	0x00800	Benutzer	Code
0x18	0x40000	0x13700	Kernel	Daten
0x20	0x3f000	0x01001	Kernel	Daten
0x28	0x80000	0x22000	Benutzer	Daten

Aufgabe 4

b) Lösen Sie die folgenden Speicherzugriffe auf. Kennzeichnen Sie potenzielle Speicherzugriffsverletzungen durch einen SEGFAULT. Die Zugriffe erfolgen, sofern nicht anders angegeben, durch einen Prozess mit Nutzerrechten.

Lesezugriff auf ss:0:

Segmentregister

cs	0x08
ss	0x30
ds	0x28
es	0x10
fs	0x18
gs	0x20

Global Descriptor Table

	Basis	Länge	Zugriff	Typ
0x0	0x10300	0x0e000	Kernel	Daten
0x8	0x10000	0x03000	Kernel	Code
0x10	0x20000	0x00800	Benutzer	Code
0x18	0x40000	0x13700	Kernel	Daten
0x20	0x3f000	0x01001	Kernel	Daten
0x28	0x80000	0x22000	Benutzer	Daten

Aufgabe 4

b) Lösen Sie die folgenden Speicherzugriffe auf. Kennzeichnen Sie potenzielle Speicherzugriffsverletzungen durch einen SEGFAULT. Die Zugriffe erfolgen, sofern nicht anders angegeben, durch einen Prozess mit Nutzerrechten.

Lesezugriff auf ss:0:

- SEGFAULT

Segmentregister

cs	0x08
ss	0x30
ds	0x28
es	0x10
fs	0x18
gs	0x20

Global Descriptor Table

	Basis	Länge	Zugriff	Typ
0x0	0x10300	0x0e000	Kernel	Daten
0x8	0x10000	0x03000	Kernel	Code
0x10	0x20000	0x00800	Benutzer	Code
0x18	0x40000	0x13700	Kernel	Daten
0x20	0x3f000	0x01001	Kernel	Daten
0x28	0x80000	0x22000	Benutzer	Daten

Aufgabe 4

b) Lösen Sie die folgenden Speicherzugriffe auf. Kennzeichnen Sie potenzielle Speicherzugriffsverletzungen durch einen SEGFAULT. Die Zugriffe erfolgen, sofern nicht anders angegeben, durch einen Prozess mit Nutzerrechten.

Lesezugriff mit Kernelrechten auf cs:0x101:

Segmentregister

cs	0x08
ss	0x30
ds	0x28
es	0x10
fs	0x18
gs	0x20

Global Descriptor Table

	Basis	Länge	Zugriff	Typ
0x0	0x10300	0x0e000	Kernel	Daten
0x8	0x10000	0x03000	Kernel	Code
0x10	0x20000	0x00800	Benutzer	Code
0x18	0x40000	0x13700	Kernel	Daten
0x20	0x3f000	0x01001	Kernel	Daten
0x28	0x80000	0x22000	Benutzer	Daten

Aufgabe 4

b) Lösen Sie die folgenden Speicherzugriffe auf. Kennzeichnen Sie potenzielle Speicherzugriffsverletzungen durch einen SEGFAULT. Die Zugriffe erfolgen, sofern nicht anders angegeben, durch einen Prozess mit Nutzerrechten.

Segmentregister

cs	0x08
ss	0x30
ds	0x28
es	0x10
fs	0x18
gs	0x20

Global Descriptor Table

	Basis	Länge	Zugriff	Typ
0x0	0x10300	0x0e000	Kernel	Daten
0x8	0x10000	0x03000	Kernel	Code
0x10	0x20000	0x00800	Benutzer	Code
0x18	0x40000	0x13700	Kernel	Daten
0x20	0x3f000	0x01001	Kernel	Daten
0x28	0x80000	0x22000	Benutzer	Daten

Lesezugriff mit Kernelrechten auf cs:0x101:

- 0x10101

Aufgabe 4

b) Lösen Sie die folgenden Speicherzugriffe auf. Kennzeichnen Sie potenzielle Speicherzugriffsverletzungen durch einen SEGFAULT. Die Zugriffe erfolgen, sofern nicht anders angegeben, durch einen Prozess mit Nutzerrechten.

Schreibzugriff auf es:0x1111:

Segmentregister

cs	0x08
ss	0x30
ds	0x28
es	0x10
fs	0x18
gs	0x20

Global Descriptor Table

	Basis	Länge	Zugriff	Typ
0x0	0x10300	0x0e000	Kernel	Daten
0x8	0x10000	0x03000	Kernel	Code
0x10	0x20000	0x00800	Benutzer	Code
0x18	0x40000	0x13700	Kernel	Daten
0x20	0x3f000	0x01001	Kernel	Daten
0x28	0x80000	0x22000	Benutzer	Daten

Aufgabe 4

b) Lösen Sie die folgenden Speicherzugriffe auf. Kennzeichnen Sie potenzielle Speicherzugriffsverletzungen durch einen SEGFAULT. Die Zugriffe erfolgen, sofern nicht anders angegeben, durch einen Prozess mit Nutzerrechten.

Schreibzugriff auf es:0x1111:

- SEGFAULT

Segmentregister

cs	0x08
ss	0x30
ds	0x28
es	0x10
fs	0x18
gs	0x20

Global Descriptor Table

	Basis	Länge	Zugriff	Typ
0x0	0x10300	0x0e000	Kernel	Daten
0x8	0x10000	0x03000	Kernel	Code
0x10	0x20000	0x00800	Benutzer	Code
0x18	0x40000	0x13700	Kernel	Daten
0x20	0x3f000	0x01001	Kernel	Daten
0x28	0x80000	0x22000	Benutzer	Daten