16.2

- Die höchste legale virtuelle Adresse im Segment 0 beträgt **19**, da 20 als Limit nicht mehr dazu gehört.
- Die niedrigste legale virtuelle Adresse in Segment 1 beträgt **108**. Dabei wird die Länge des Adressraumes mit dem Limit subtrahiert.

  128 – 20 = 108

- Die niedrigste illegale physische Adresse ist 20, da die physische Adresse von Segment 0 von **0 bis 19** geht.
- Die höchste illegale physische Adresse ist 491. Dies kommt daher, da die physische Adresse für Segment 1 von **492 bis 511** geht. Es geht nicht bis 512, da 512 das Limit ist und nicht mitinbegriffen ist. Das Limit ist 20, daher geht es auch nur bis 492.

Folgendermaßen würde ich segmentation.py laufen lassen mit -A um es zu testen:
./segmentation.py -a 128 -p 512 -b 0 -l 20 -B 512 -L 20 -A 19,20,107,108 -c
Bei der Ausführung kann man gut sehen, dass 19 akzeptiert wird 20 hingegen nicht, da es illegal ist. Dasselbe sieht man bei 107 und 108. 108 ist legal, während bei 107 ein Segmentation Fault auftritt.

16.3

Die Aufgabe ist, dass wir die Base-Bound Paare für Segment 0 und Segment 1 finden sollen. Hierfür ist gegeben, dass wir einen 16 Byte Adressraum haben und 128 Byte physikalischen Speicher. Auch ist eine Liste, von 0 – 15 möglichen virtuellen Adressen gegeben. Hier soll beachtet werden, dass nur 0,1 & 14,15 valid sein sollen. Hierfür werden diese in binär übersetzt und auf 4 Stellen aufgefüllt da wir uns in einem 16 Byte Adressraum befinden:

0: 0000

1: 0001

14: 1110

15: 1111

Da der erste Bit das Topbit ist, gibt es an in welchem Modul die virtuelle Adresse ist. Daraus ergibt sich, dass 0 & 1 in Segment 0 sein müssen und 14 & 15 in Segment 1.

Für Segment 0 schauen wir also die binär zahlen ohne Topbit an: 000 & 001.
Daraus ergibt sich 2, denn 0 < 2 & 1 < 2.
Für Segment 1 schauen wir uns auch die binär zahlen ohne Topbit an: 110 & 111. Dies wird dann wieder zu Dezimal übersetzt: 6 & 7.

Dann wird die Max Segmentgröße berechnet: 2^[log2(16) anzahl segmentation bits] = 2^(4 – 1) = 8

Dadurch kann man den Offset berechnen: 6 – 8 = |-2| & 7 – 8 = |-1|

Daraus ergibt sich dann ein Limit von 2: |-2| |-1| <= 2

Segment 0: base: 0 limit: 2 (geht also von 0 – 1)
Segment 1: base: 128 limit: 2 (geht also von 128 – 127)

16.4

Um dieses Problem zu lösen, muss die Größe des virtuellen Adressraumes 90% der Größe des physischen Arbeitsspeichers entsprechen.

Dies kann man durch folgende Flags ermöglichen:
./segmentation.py **-a 90 -p 100** -c

16.5

./segmentation.py -b 1 -l 1 -B 1 -L 1 -c

Segment 1 und 2 sollen als Base- und Boundwert jeweils denselben Wert haben (in unserem Beispiel eine 1). Da der Basewert nicht berücksichtigt wird, ist eine 1 nicht gültig. Somit ist keine einzige zufällig generierte virtuelle Adresse gültig.

16.6 - Zusatzaufgabe

Segment0 | Base	0
Segment0 | Limit	64
Segment1 | Limit	128
Segment1 | Base	364

Die Größe der virtuellen Adresse ist oben einsehbar (ARG Größe des Adressraumes von 364).

Dezimal	Binär	Segment	If Segment1-erste Binärstelle ignorieren	Offset	Valid?
1	00000001	Segment 0		1	Ja
17	00010001	Segment 0		17	Ja
33	00100001	Segment 0		33	Ja
55	00110111	Segment 0		55	Ja
67	01000011	Segment 0		67	Nein
80	01010000	Segment 0		80	Nein
92	01011100	Segment 0		92	Nein
108	01101100	Segment 0		108	Nein
122	01111010	Segment 0		122	Nein
132	10000100	Segment 1	100=4 | 4 – 128	-124	Ja
180	10110100	Segment 1	110100=52 | 52 – 128	-76	Ja
207	11001111	Segment 1	1001111=79 | 79 – 128	-49	Ja
233	11101001	Segment 1	1101001=105 | 105 - 128	-23	Ja
255	11111111	Segment 1	1111111=127 | 127 - 128	-1	Ja
299	100101011	Segment 1	101011=43 | 43 - 256	-213	Nein
332	101001100	Segment 1	1001100=76 | 76 - 256	-180	Nein

Als erstes wandelt man die Dezimalzahl in Binär um. Im Anschluss wird die Binärzahl aufgefüllt mit Nullen bis zur 8ten Binärstelle, da wir hier von einer virtuellen Adressgröße von 256 Byte ausgegangen sind. Die Dezimalzahlen 299 und 332 werden mit 9 Binärstellen berechnet, da wir hier von einer virtuellen Adressraumgröße von 512 Byte ausgehen. Da die Zahlen von sich aus bereits 9 Stellen haben, muss hier nicht mit Nullen aufgefüllt werden.

Nun sagt die vorderste Binärstelle aus, ob die Adresse sich im Segment 1 oder 0 befindet.

Falls die Adresse sich im Segment 0 befindet, haben wir auch bereits den Offset gefunden. Hierfür können wir direkt vergleichen, ob die Adresse sich innerhalb des Limits von Segment 0 befindet mit dem Wert 64. Ist also eine Zahl im Segment 0 und hat einen Wert von unter 64 ist sie innerhalb des virtuellen Adressraums von Segment 0.

Bei Segment 1 müssen wir noch einen Rechenschritt hinzufügen. Die erste Stelle der Binärzahl wird weggestrichen, da diese lediglich das Segment beschreibt. Die „neue“ Binärzahl ohne 1 am Anfang wird nun erneut umgewandelt. Der Wert wiederrum wird mit der maximalen Segmentgröße subtrahiert ohne das Topbit (bei den Werten mit 8 Stellen also 2^7 und bei den Werten mit 9 Stellen 2^8).

Der Wert wird als Betrag (z.B. |-5| = 5) mit dem Limit verglichen von Segment 1. Liegt der Wert darunter ist die virtuelle Adresse gültig, andernfalls nicht.