Question 1

First let's use a tiny address space to translate some addresses. Here's a simple set of parameters with a few different random seeds; can you translate the addresses?

segmentation.py -a 128 -p 512 -b 0 -l 20 -B 512 -L 20 -s 0
segmentation.py -a 128 -p 512 -b 0 -l 20 -B 512 -L 20 -s 1
segmentation.py -a 128 -p 512 -b 0 -l 20 -B 512 -L 20 -s 2

Offset = remaining bits in dec. - Max. Segment
Boolean (Valid?) = |absolut offset| <= limit
Physical Address = Base + offset
segment bit: (log2(address space size)-1)

ARG seed 0
ARG address space size 128
ARG phys mem size 512

Segment register information:

Segment 0 base (grows positive) : 0x00000000 (decimal 0)
Segment 0 limit : 20

Segment 1 base (grows negative) : 0x00000200 (decimal 512)
Segment 1 limit : 20

Virtual Address Trace
VA 0: 0x0000006c (decimal: 108) --> PA or segmentation violation?
VA 1: 0x00000061 (decimal: 97) --> PA or segmentation violation?
VA 2: 0x00000035 (decimal: 53) --> PA or segmentation violation?
VA 3: 0x00000021 (decimal: 33) --> PA or segmentation violation?
VA 4: 0x00000041 (decimal: 65) --> PA or segmentation violation?

segment bit: (log2(address space size)-1) = 6
Max. Segment = 2^[log2(address size) - # of segment bits] = 2^6 = 64

Segment 1 example:
108 = 1101100 => 1 = seg 1

Offset = remaining bits in dec. - Max. Segment = 101100 - 64 = 44 - 64 = -20
Boolean (Valid?) = |absolut offset| <= limit = |20| <= 20 = true
Physical Address = Base + offset = 512 + (-20) = 492

Segment 0 example:
53 = 0110101 => 0 = seg 0

Offset = remaining bits in dec. = 110101 = 53
Boolean (Valid?) = |absolut offset| <= limit = |53| <= 20 => false
Physical Address = Base + offset = SEGV

dec. adress	Seg	Offset	Valid	Physical address
108 = 1101100	1	44 - 64 = -20	|-20| <= 20 => true	512 + (-20) = 492
97 = 1100001	1	33 - 64 = -31	|-31| <= 20 => false	SEGV
53 = 0110101	0	53	|53| <= 20 => false	SEGV
33 = 0100001	0	33	|33| <= 20 => false	SEGV
65 = 1000001	1	1 - 64 = -63	|-63| <= 20 => false	SEGV

Offset = remaining bits in dec. - Max. Segment
Boolean (Valid?) = |absolut offset| <= limit
Physical Address = Base + offset
segment bit: (log2(address space size)-1)
Max. Segment = 2^[log2(address size) - # of segment bits]

ARG seed 1
ARG address space size 128
ARG phys mem size 512

Segment 0 base (grows positive) : 0x00000000 (decimal 0)
Segment 0 limit : 20

Segment 1 base (grows negative) : 0x00000200 (decimal 512)
Segment 1 limit : 20

Virtual Address Trace
VA 0: 0x00000011 (decimal: 17) --> PA or segmentation violation?
VA 1: 0x0000006c (decimal: 108) --> PA or segmentation violation?
VA 2: 0x00000061 (decimal: 97) --> PA or segmentation violation?
VA 3: 0x00000020 (decimal: 32) --> PA or segmentation violation?
VA 4: 0x0000003f (decimal: 63) --> PA or segmentation violation?

segment bit: 6
Max. Segment = 64

dec. adress	Seg	Offset	Valid	Physical address
17 = 0010001	0	17	|17| <= 20 => true	0 + 17 = 17
108 = 1101100	1	44 - 64 = -20	|-20| <= 20 => true	512 + (-20) = 492
97 = 1100001	1	33 - 64 = -31	|-31| <= 20 => false	SEGV
32 = 0100000	0	32	|32| <= 20 => false	SEGV
63 = 0111111	0	63	|63| <= 20 => false	SEGV

Offset = remaining bits in dec. - Max. Segment
Boolean (Valid?) = |absolut offset| <= limit
Physical Address = Base + offset
segment bit: (log2(address space size)-1)
Max. Segment = 2^[log2(address size) - # of segment bits]

ARG seed 2
ARG address space size 128
ARG phys mem size 512

Segment register information:

Segment 0 base (grows positive) : 0x00000000 (decimal 0)
Segment 0 limit : 20

Segment 1 base (grows negative) : 0x00000200 (decimal 512)
Segment 1 limit : 20

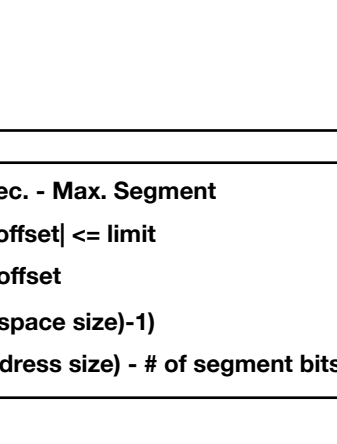
Virtual Address Trace
VA 0: 0x0000007a (decimal: 122) --> PA or segmentation violation?
VA 1: 0x00000079 (decimal: 121) --> PA or segmentation violation?
VA 2: 0x00000007 (decimal: 7) --> PA or segmentation violation?
VA 3: 0x0000000a (decimal: 10) --> PA or segmentation violation?
VA 4: 0x0000006a (decimal: 106) --> PA or segmentation violation?

segment bit: 6
Max. Segment = 64

dec. adress	Seg	Offset	Valid	Physical address
122 = 1111010	1	58 - 64 = -6	|-6| <= 20 => true	512 + (-6) = 506
121 = 1111001	1	57 -64 = -7	|-7| <= 20 => true	512 + (-7) = 505
7 = 0000111	0	7	|7| <= 20 => true	0 + 7 = 7
10 = 0001010	0	10	|10| <= 20 => true	0 + 10 = 10
106 = 1101010	1	42 - 64 = -22	|-22| <= 20 => false	SEGV

Question 2

Now, let's see if we understand this tiny address space we've constructed (using the parameters from the question above).
What is the highest legal virtual address in segment 0?
What about the lowest legal virtual address in segment 1?
What are the lowest and highest *illegal* addresses in this entire address space?
Finally, how would you run segmentation.py with the -A flag to test if you are right?



Highest legal virtual address in seg 0:
19, since the limit is 20 and if you count the 0 you get 20 addresses (growing pos.)
Lowest legal virtual address in seg 1:
128 - 20 = 108 (growing neg.)
Highest illegal virtual address in the entire address space:
108 -1 = 107 (growing neg.)
Lowest illegal virtual address in the entire address space:
19 +1 = 20 (growing pos.)

Question 3

Let's say we have a tiny 16-byte address space in a 128-byte physical memory. What base and bounds would you set up so as to get the simulator to generate the following translation results for the specified address stream: valid, valid, violation, ..., violation, valid, valid? Assume the following parameters:

segmentation.py -a 16 -p 128 -A 0,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15 --b0 ? --l0 ? --b1 ? --l1 ?

Segment 0 base: 0 (first address on stack)
Segment 0 bound: 2 (size)
Segment 1 base: 128 (first address on heap)
Segment 1 bound: 2 (size)

segmentation.py -a 16 -p 128 -A 0,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15 --b0 **0** --l0 **2** --b1 **128** --l1 **2**

Question 4

Assume we want to generate a problem where roughly 90% of the randomly-generated virtual addresses are valid (not segmentation violations). How should you configure the simulator to do so? Which parameters are important to getting this outcome?

To achieve this, 90% of all possible addresses need to be within the bounds.
0.9 * 128/2 = 57,6 => 57
python3 segmentation.py -a 128 -p 512 --b0 0 --l0 57 --b1 512 --l1 57 -c

Question 5

Can you run the simulator such that no virtual addresses are valid? How?

Just set bound to 0
python3 segmentation.py -a 128 -p 512 --b0 0 --l0 0 --b1 512 --l1 0 -c