```
*************************************************************************
```
Authors : Albert Owusu-Asare , Box 4497, <owusuasa@grinnell.edu>
: Mark Lewis , Box 3948, <lewismar@grinnell.edu>

This document contains written answers to lab 4: Memory
All questions are answered except those marked [Exploration].

Questions:
http://www.cs.grinnell.edu/~weinman/courses/CSC213/2014F/labs/memory.html

```
*************************************************************************
```

PART A : Address Translation

2.
VA 0: 64 -> 12949
VA 1: 209 -> 13094
VA 2: 1 -> 12886
VA 3: 862 -> 13146
VA 4: 862 -> Invalid

3. The bound must be 930, that is, the highest decimal value, plus one.

4. 2^10.

PART B: Segmentation

1.
a.
108 -> 492 Valid
97 -> invalid
53 -> invalid
33 -> invalid
65 -> invalid
b. The highest legal virtual address is 19 because we have the virtual space is 20 units long, starting at number 0.
c. The lowest legal virutal address in segment 1 is 108, or 128 (the base of our virtual space) - 20 (the bound of our virutal space).
d.The lowest illegal address is 20, and the highest is 491. This is because in the real address space, segment 0 starts at 0 and continues until 19 (thus 20 is illegal). 107 is the highest illegal virtual address, and it corresponds to the physical address 491.

2.
VA 0: 124 -> 508 valid segment 1
VA 1: 9 -> 9 valid segment 0
VA 2: 43 -> invalid inbetween segments
VA 3: 67 -> invalid inbetween segments
VA 4: 31 -> invalid inbetween segments
VA 5: 17 -> 17 valid segment 0
VA 6: 92 -> invalid inbetween segments
VA 7: 112 -> 496 valid segment 1
VA 8: 76 -> invalid inbetween segments
VA 9: 43 -> invalid inbetween segments

3. This problem was completed with the help of Professor Weinman.
We want the limit to be two in each case so that we can have two adjacent bytes be valid in a segment. The bases, then, go whever they can fit. An easy way to do this, would be b0 = 0, l0 = 2, b1 = 16, l1 = 2

4. To ensure that approximately 90% of the virtual address space is valid, we need to pay attention to three things: the total size of the virtual space, the size of the base/bounds pair of segment 0, and the size of the base/bound pair of segment 1. To do this, the combined space of segment 0 and segment 1 needs to be approximately 90% of the total virtual address space.

PART C: Linear Paging
---

1. We expect the size of the page table to change directly with the size of the address space. For example, if we have a page table with 8 pages, decreasing the address space size in half and leaving all other values constant will cause the size of the page table to be cut in half (have 4 pages).

3. We expect the the size of a page and the size of a page table (the number of pages) to be inversely related. This is because we have a finite address space, and we fit as many pages of the page size into the space as possible. Therefore, double the size of a page will cut in half the amount of pages that that can be stored in the address space.

6. This problem was completed with the help of Professor Weinman.
VA 0x00000630 (decimal: 1584): VPN 0011 -> 3, offset 0000110000 -> 48
in page 3, we start at hex 2, or,
0010, so together, the physical address is
00100000110000
VA 0x0000363d (decimal: 13885): VPN 1101 -> 13, offset 1000111101
in page 13, start at hex 4, or,
0100. Together, the physical address is
01001000111101.
VA 0x000026a3 (decimal: 9891) VPN 1001 -> 9, which is not valid
VA 0x000033a7 (decimal: 13223) VPN 1100 -> 12, offset 1110100111
in page 12, we have hex 16 or
10110. Together, the physical address
101101110100111
VA 0x00002eb3 (decimal: 11955) VPN 1011 -> 11, which is invalid