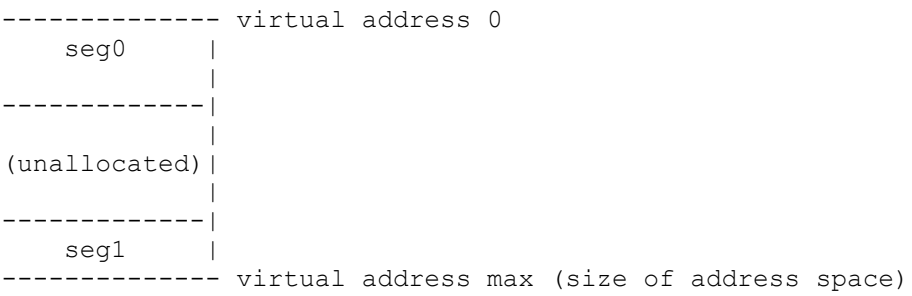
segmentation.py

This program allows you to see how address translations are performed in a system with segmentation. The segmentation that this system uses is pretty simple: an address space has just *two* segments; further, the top bit of the virtual address generated by the process determines which segment the address is in: 0 for segment 0 (where, say, code and the heap would reside) and 1 for segment 1 (where the stack lives). Segment 0 grows in a positive direction (towards higher addresses), whereas segment 1 grows in the negative direction.



With segmentation, as you might recall, there is a base/limit pair of registers per segment. Thus, in this problem, there are two base/limit pairs. The segment-0 base tells which physical address the *top* of segment 0 has been placed in physical memory and the limit tells how big the segment is; the segment-1 base tells where the *bottom* of segment 1 has been placed in physical memory and the corresponding limit also tells us how big the segment is (or how far it grows in the negative direction).

To run with the default flags, type either of the following:
```
prompt> ./segmentation.py
```

You should see this:
```
  ARG seed 0
  ARG address space size 1k
  ARG phys mem size 16k

  Segment register information:
    Segment 0 base  (grows positive) : 0x00001aea (decimal 6890)
    Segment 0 limit                  : 472

    Segment 1 base  (grows negative) : 0x00001254 (decimal 4692)
    Segment 1 limit                  : 450

  Virtual Address Trace
    VA  0: 0x0000020b (decimal:  523) --> PA or segmentation violation?
    VA  1: 0x0000019e (decimal:  414) --> PA or segmentation violation?
    VA  2: 0x00000322 (decimal:  802) --> PA or segmentation violation?
    VA  3: 0x00000136 (decimal:  310) --> PA or segmentation violation?
    VA  4: 0x000001e8 (decimal:  488) --> PA or segmentation violation?
```

For each virtual address, either write down the physical address it translates to OR write down that it is an out-of-bounds address (segmentation violation). For this problem, you should assume a simple address space with two segments: the top bit of the virtual address can be used to check whether the virtual address is in segment 0 (topbit=0) or segment 1 (topbit=1). Note that the base/limit pairs given to you grow in different directions, depending on the segment (seg 0 grows in the positive direction, seg 1 negative).