On my honor, I have not given, nor received, nor witnessed any unauthorized assistance on this work.

Print name and sign: _____

Question:	1	2	3	Total
Points:	12	8	10	30
Score:				

1. (6 points) When we virtualize memory, we have three primary goals (listed below). Provide a (brief!) explanation of each goal with respect to memory virtualization.

   (a) (2 points) transparency

   Solution: "The OS should implement virtual memory in a way that is invisible to the running program." OSTEP 13.4

   (b) (2 points) efficiency

   Solution: "The OS should strive to make the virtualization as efficient as possible, both in terms of time (i.e., not making programs run much more slowly) and space (i.e., not using too much memory for structures needed to support virtualization). "

   (c) (2 points) protection

   Solution: "The OS should make sure to protect processes from one another as well as the OS itself from processes. When one process performs a load, a store, or an instruction fetch, it should not be able to access or affect in any way the memory contents of any other process or the OS itself (that is, anything outside its address space)."

2. Assume a system is using base and bounds with the following system characteristics:

   - a 1KB (1024 bytes) virtual address space
   - a base register set to 10000
   - a bounds register set to 100

   For each of the **virtual addresses**, give the corresponding physical memory location or state that it would generate a fault.

   (a) (2 points) 0 ___10000___

   (b) (2 points) 99 ___10099___

   (c) (2 points) 100 ___Fault___

   (d) (2 points) 1000 ___Fault___

3. Consider a system uses segmentation and has 2 segments:

   - segment 0 for code and a positive growing heap (grows towards higher addresses)
   - segment 1 for a negatively-growing stack (grows towards smaller addresses).

   This system has a virtual address space size of only 128 bytes, and there is only 1K of physical memory.

   Below is a memory trace from a program. In particular, the trace tells you which virtual address (VA) was accessed (in both hex and decimal), and then whether or not the access was valid or not, and what segment the virtual address tried to access (thus, you **do not** need to map these addresses to segments yourself).

   If valid, the physical address (in both hex and decimal) are reported. Magically, this program kept running after memory-access violations, and thus we have a long trace.

```
VA 0x0000006c (decimal: 108) --> VALID in SEG1: 0x000003ec (decimal: 1004)
VA 0x0000001d (decimal: 29) --> VALID in SEG0: 0x0000021d (decimal: 541)
VA 0x00000050 (decimal: 80) --> SEGMENTATION VIOLATION (SEG1)
VA 0x0000001e (decimal: 30) --> SEGMENTATION VIOLATION (SEG0)
VA 0x00000058 (decimal: 88) --> VALID in SEG1: 0x000003d8 (decimal: 984)
VA 0x00000061 (decimal: 97) --> VALID in SEG1: 0x000003e1 (decimal: 993)
VA 0x00000035 (decimal: 53) --> SEGMENTATION VIOLATION (SEG0)
VA 0x00000021 (decimal: 33) --> SEGMENTATION VIOLATION (SEG0)
VA 0x00000064 (decimal: 100) --> VALID in SEG1: 0x000003e4 (decimal: 996)
VA 0x0000003d (decimal: 61) --> SEGMENTATION VIOLATION (SEG0)
VA 0x0000000c (decimal: 12) --> VALID in SEG0: 0x0000020c (decimal: 524)
VA 0x00000005 (decimal: 5) --> VALID in SEG0: 0x00000205 (decimal: 517)
VA 0x0000002f (decimal: 47) --> SEGMENTATION VIOLATION (SEG0)
```

   (a) (2 points) From this trace, what was the base register of segment 0 set to? 512 or 0x400

   Solution: To calculate this, find a valid reference for segment 0. For example:
   `VA 0x00000005 (decimal: 5) --> VALID in SEG0: 0x00000205 (decimal: 517)`

   Here you can see that virtual address 5 translates to physical address 517. Subtracting the offset into segment0 from 517 gets us 512, which must be the base address (0x200).

   (b) (2 points) From this trace, what was the bounds register of segment 0 set to? 30

   Solution: For this, you need to find (in the best case) two references, one that is at virtual address N and is valid, and one that is at address N+1 but is not valid; that will tell us the bound precisely. Fortunately, there are two such references:

```
VA 0x0000001d (decimal: 29) --> VALID in SEG0: 0x0000021d (decimal: 541)
VA 0x0000001e (decimal: 30) --> SEGMENTATION VIOLATION (SEG0)
```

This tells us that any virtual address from 0 through 29 is valid, whereas 30 is not. Hence, the size of segment zero is 30, which we thus conclude is the bound.

Common mistakes: listing an address instead of size (and reporting the value in the bounds register as 542). Remmber that the bounds register contains a SIZE not an ADDRESS.

(c) (2 points) From this trace, what was the base register of segment 1 set to? _____

Solution: Similar to the question above, but remember that segment 1 goes backwards:

```
VA 0x0000006c (decimal: 108) --> VALID in SEG1: 0x000003ec (decimal: 1004)
```

Thus, if 108 maps to 1004, we need to figure out what V.A. 127 (the last valid address of our address space) would map to.

We know that we need to add 19 to 108 to get 127. Thus, we need to add 19 to the physical address 1004 to get the last valid address of the negative growing stack segment. Thus, V.A 127 maps to physical address 1023. Hence, the base would be 1024, the next address beyond the end of where the backward-growing segment lies.

Common mistake: moving the wrong way in memory and subtracting 19 from 1004 to get 985 (and setting the base register value to 986).

(d) (2 points) From this trace, what was the bounds register of segment 1 set to?

Hint: You cannot state a precise value. Give a range. $40 \leq x \leq 47$

Solution: The bound here is harder, but the two relevant references are:

```
VA 0x00000050 (decimal: 80) --> SEGMENTATION VIOLATION (SEG1)
VA 0x00000058 (decimal: 88) --> VALID in SEG1: 0x000003d8 (decimal: 984)
```

As you can see, 80 goes too far backwards, but 88 is fine. With 88, the size must at least allow 88 through 127 to be valid; thus 40 bytes is the minimum segment size. It clearly could also be that 81 is valid, which adds 7 more bytes into the segment size, or 47. Thus, the size of this segment is between 40 and 47, inclusive.

(e) (2 points) The mechanism of segmentation often leads to **external fragmentation**. Explain what fragmentation is and how it occurs.

Solution: OSTEP: Ch. 16, pg. 9

When we allocate small chunks of memory, we quickly end up with a memory that's full of small holes. These holes may not be adequate in size for another process's needed size.