quiz 4 submission form

The respondent's email address (**dmitryvengertsev@u.boisestate.edu**) was recorded on submission of this form.

Your name *

Dmitry Vengertsev

problem 1

Virtual address space vs physical memory space, which one is bigger? (5 points)

Virtual address space can be bigger or smaller than physical address space. Even though it was initially not intuitive for me that virtual space can be bigger, the trick is that virtual address space can use both RAM and hard disk space.

problem 2

Here's a little program that prints out the locations of the main() routine (where code lives), the value of a heap-allocated value returned from malloc(), and the location of an integer on the stack:

```
#include <stdio.h>
#include <stdlib.h>
```

```
int main(int argc, char *argv[]) {
    printf("location of code : %p\n", main);
    printf("location of heap : %p\n", malloc(100e6));
    int x = 3;
    printf("location of stack: %p\n", &x);
    return 0;
}
```

When run on a 64-bit Mac, we get the following output:
location of code : 0x1095afe50
location of heap : 0x1096008c0
location of stack: 0x7fff691aea64

In the output of the above program, is this address - 0x1095afe50, a virtual address or a physical address? (5 points)

- (●) A. virtual address
- ( ) B. physical address

In the output of the above program, is this address - 0x1096008c0, a virtual address or a physical address? (5 points)

- (●) A. virtual address
- ( ) B. physical address

In the output of the above program, is this address - 0x7fff691aea64, a virtual address or a physical address? (5 points)

- (●) A. virtual address
- ( ) B. physical address

problem 3

Address Translation: Segmentation.

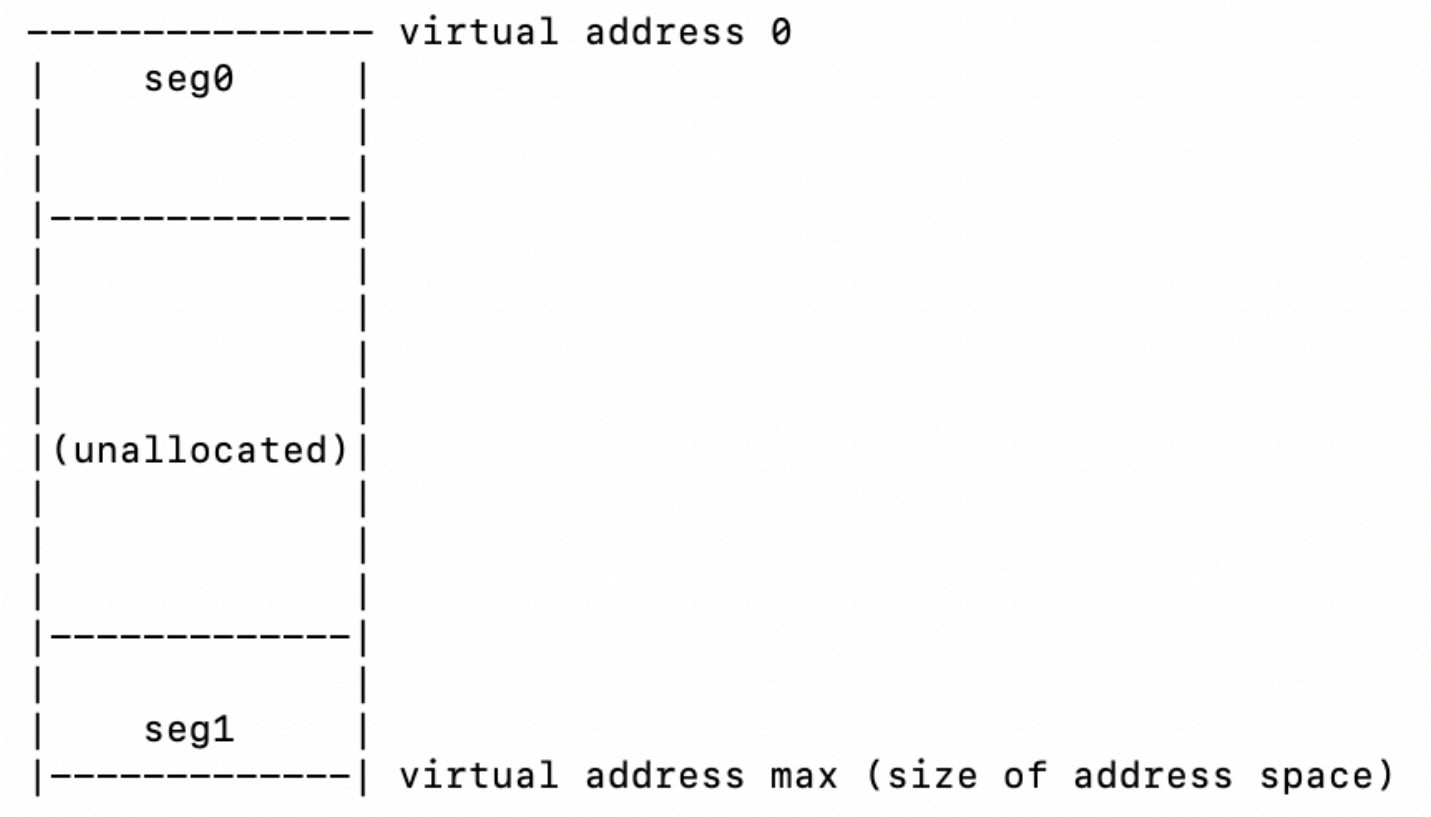
Suppose we have the following setup:
address space size: 128
phys mem size: 256

The address visually looks like the picture below. The address space has two segments:

Segment register information:
Segment 0 base (grows positive): 0x00000000 (decimal 0)
Segment 0 limit   : 40
Segment 1 base (grows negative): 0x00000100 (decimal 256)
Segment 1 limit   : 40

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).

Question: For each virtual address below, either write down the physical address it translates to OR write down that it is an out-of-bounds address (a segmentation violation).



VA 0: 0x00000065 (decimal: 101) --> PA or segmentation violation? (5 points)

Valid in seg1: decimal: 229

VA 1: 0x00000069 (decimal: 105) --> PA or segmentation violation? (5 points)

Valid in seg1: decimal: 233

VA 2: 0x0000003e (decimal: 62) --> PA or segmentation violation? (5 points)

Segmentation violation

VA 3: 0x00000021 (decimal: 33) --> PA or segmentation violation? (5 points)

Valid in seg0: decimal: 32

problem 4

Address Translation: Paging.

A simulation trace is generated below using the tool paging-linear-translate.py.
ARG address space size 8k
ARG phys mem size 32k
ARG page size 1k

The format of the page table is simple:
The high-order (left-most) bit is the VALID bit.
  If the bit is 1, the rest of the entry is the PFN.
  If the bit is 0, the page is not valid.

Page Table (from entry 0 down to the max size)
  [0] 0x80000018
  [1] 0x00000000
  [2] 0x00000000
  [3] 0x8000000c
  [4] 0x80000009
  [5] 0x00000000
  [6] 0x8000001d
  [7] 0x80000013

Question: For each virtual address below, write down the physical address it translates to

OR write down that it is invalid.

VA 0x00000803 (decimal: 2051) --> PA or invalid address? (10 points)

Invalid

VA 0x00001d1b (decimal: 7451) --> PA or invalid address? (10 points)

19739

VA 0x000019ec (decimal: 6636) --> PA or invalid address? (10 points)

30188

VA 0x00001cde (decimal: 7390) --> PA or invalid address? (10 points)

19678

problem 5

Address Translation: Paging.

A simulation trace is generated below using the tool paging-linear-translate.py.

ARG address space size 128
ARG phys mem size 512

ARG page size 8

The format of the page table is simple:
The high-order (left-most) bit is the VALID bit.
  If the bit is 1, the rest of the entry is the PFN.
  If the bit is 0, the page is not valid.
Use verbose mode (-v) if you want to print the VPN # by
each entry of the page table.

Page Table (from entry 0 down to the max size)
  [0] 0x8000002f
  [1] 0x8000003c
  [2] 0x8000003b
  [3] 0x00000000
  [4] 0x00000000
  [5] 0x80000029
  [6] 0x80000007
  [7] 0x00000000
  [8] 0x00000000
  [9] 0x80000024
  [10] 0x00000000
  [11] 0x00000000
  [12] 0x00000000
  [13] 0x80000031
  [14] 0x00000000
  [15] 0x80000008

Question: For each virtual address below, write down the physical address it translates to
OR write down that it is invalid.

VA 0x0000004f (decimal: 79) --> PA or invalid address? (10 points)

Invalid

VA 0x0000001c (decimal: 28) --> PA or invalid address? (10 points)

Invalid