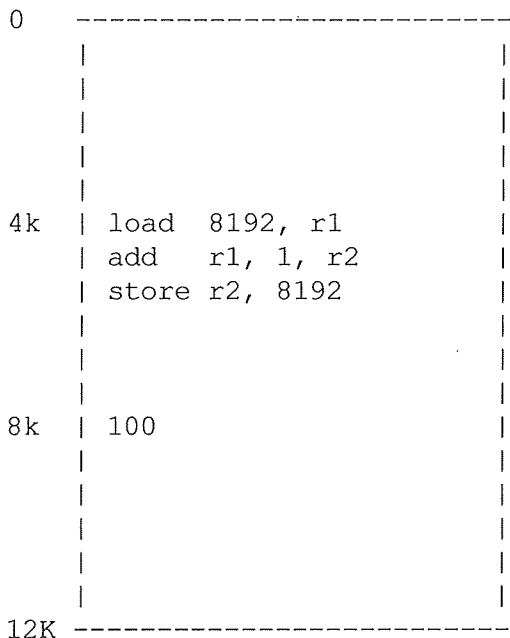Here is some assembly code:

```
load  8192 r1    # load value at memory location 8192 -> r1
add   r1, 1, r2  # put r1 + 1 -> r2
store r2, 8192   # stores r2 -> memory location 8192
```

Assume each instruction takes up 4 bytes in memory.

Assume the program counter (PC) is set to 4096 (4k) when running the first instruction of this sequence. The virtual address space of this process looks like this (not to scale):

```
0     ------------------------------
      |                            |
      |                            |
      |                            |
      |                            |
      |                            |
4k    | load   8192, r1            |
      | add    r1, 1, r2           |
      | store  r2, 8192            |
      |                            |
      |                            |
      |                            |
8k    | 100                        |
      |                            |
      |                            |
      |                            |
      |                            |
      |                            |
12K   ------------------------------
```

The total size of this virtual address space is 12 KB.

Assume this is a system with "base and bounds" registers used for memory relocation and protection.

In this example, assume that the process's address space is loaded into physical memory at physical address 64 KB (this is the base). The bounds is set to the size of the address space: 12 KB.

List all the physical memory locations that are referenced during the execution of this three-instruction sequence.