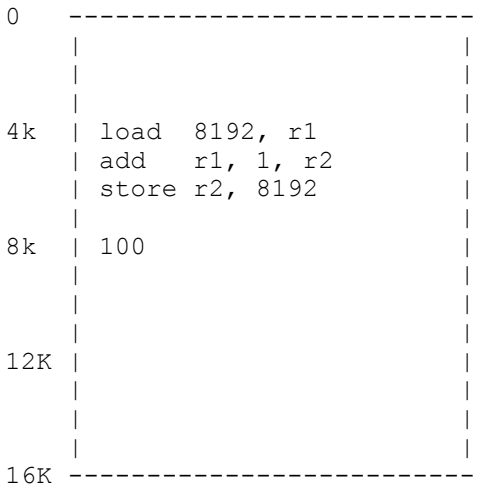
Here is some assembly code:

```
load  8192, r1   # loads value at memory (8192) -> r1
add   r1, 1, r2  # adds 1 to r1; result into r2
store r2, 8192   # stores r2 into memory (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):



Assume this is a system with a hardware-managed, linear page table.
The total size of this virtual address space is 16 KB.
The page size for the system is 1 KB.
Physical memory is of size 64 KB.

Each page table entry (PTE) looks like this:

valid	page frame number	protection bit
1 bit	6 bits	1 bit

(valid: 1->page is valid; protection: 1->read/write, 0->read only)

The PTBR for this process points to a physical address where the process's page table is located: 32KB.

The contents for this page table are:

```
0x00 0x00 0x00 0x00 0x8A 0x00 0x00 0x00 0xA9 0x00 0x00 0x00 0x00 0x00 0x00 0x00
```

Your task: List all the physical memory locations that are referenced during the execution of this three-instruction sequence, first assuming there is NO TLB, then assuming there is a TLB of infinite size.