Some basic assumptions:
- The page size is an unrealistically-small 32 bytes
- The virtual address space for the process in question (assume there is only one) is 1024 pages, or 32 KB
- physical memory consists of 128 pages

Thus, a virtual address needs 15 bits (5 for the offset, 10 for the VPN).
A physical address requires 12 bits (5 offset, 7 for the PFN).

The system assumes a multi-level table. Thus, the upper five bits of a virtual address are used to index into a directory; the directory entry (PDE), if valid, points to a of the table. Each table holds 32 page-table entries (PTEs). Each PTE, if valid, holds the desired translation (physical frame number, or PFN)
of the virtual in question.

The format of a PTE is thus:
VALID | PFN6 ... PFN0

and is thus 8 bits or 1 byte.

The format of a PDE is essentially identical:
VALID | PT6 ... PT0

You are given two pieces of information to begin with.

First, you are given the value of the directory base register (PDBR), which tells you which the directory is located upon.

Second, you are given a complete dump of each of memory. A dump looks like this:

```
0: 08 00 01 15 11 1d 1d 1c 01 17 15 14 16 1b 13 0b ...
1: 19 05 1e 13 02 16 1e 0c 15 09 06 16 00 19 10 03 ...
2: 1d 07 11 1b 12 05 07 1e 09 1a 18 17 16 18 1a 01 ...
...
```

which shows the 32 bytes found on pages 0, 1, 2, and so forth. The first byte (0th byte) on 0 has the value 0x08, the second is 0x00, the third 0x01, and so forth.

Use the PDBR to find the relevant table entries for this virtual page. Then find if it is valid. If so, use the translation to form a final physical address. Using this address, you can find the VALUE that the memory reference is looking for.

Of course, the virtual address may not be valid and thus generate a fault.