Suppose we have a system with a page size of 1K and a 4K virtual address space. The physical memory size is 16K.

A single load instruction is stored at virtual address 1024 (1K):

```
ld 2048, r1
```

This instruction generates two memory accesses: an instruction fetch from virtual address 1024 and a data load from virtual address 2048.

1. Draw the virtual address space. Include overall size (starting and ending addresses), load instruction and data (you can make up a value if you want to).

Solution:

```
------------------ 0
|                |
|                |
|                |
----------------- 1024, 2K
| ld 2048, r1    |          on a 32 bit machine, instr is
----------------- 1028      4 bytes
|                |
|                |
|                |
------------------ 2048, 2K
| data here      |          size isn't specified in the problem but
------------------ 2052     an int would be 4 bytes
|                |
|                |
|                |
------------------ last valid address is 4095 (4k == 4096)
```

2. How many pages are in the virtual address space? Add those to your previous drawing.

Solution: 4K total address space / 1K page size == 4 pages

```
 ---> ------------------ 0
  |    |                |
 VP0   |                |
  |    |                |
 ---> ------------------ 1024, 1K
  |    | ld 2048, r1    |
  |    |----------------- 1028
 VP1   |                |
  |    |                |
  |    |                |
 ---> ----------------- 2048, 2K
  |    | data           |
  |    ----------------- 2052
 VP2   |                |
  |    |                |
  |    |                |
 ---> ----------------- 3096, 3K
  |    |                |
 VP3   |                |
  |    |                |
 ---> ------------------ last valid address is 4095 (4k == 4096)
```

3. How many bits are needed to encode each virtual address?

Solution: We need to access every byte in 4k address space. How many bits to represent 4096 byte's addresses? $2^x == 4096$, x = 12 bits.

4. Show how to divide the virtual address into a virtual page number and offset.

Solution: Need to differentiate between 4 pages in address space: $2^x == 4$ so need x = 2 bits for this portion.
Need to be able to address 1024 individual bytes in a page: $2^x == 1024$ so need x = 10 bits for this portion.
So need 12 bits total, divided like:

```
|x|x|x|x|x|x|x|x|x|x|x|x|
^ VPN ^      Offset     ^
|-----|-----------------|
```

5. How many bits are needed to encode each physical address?

> **Solution:** One way to think about it:
> Need to access every byte in 16K address physical memory. 2^x == 16384, x == 14 bits.
>
> Alternate reasoning:
> 16 physical pages means we need 4 bits to represent the PFN + 10 bit offset from virtual address == 14 bits total.

6. Now suppose the system has a (hardware managed) linear page table. Each page table entry (PTE) consists of 8 bits:

```
| valid | Read  | Write | eXecute |  PFN  |
| 1 bit | 1 bit | 1 bit | 1 bit   | 4 bits|
```

The RWX bits are set to 1 if reading, writing, and execution respectively are allowed on a page.
The 4 entries in the page table are:
0xF5
0xCA
0xFF
0x00

(a) Find the physical address of the INSTRUCTION.

(b) Find the physical address of the DATA.

> **Solution:** Note that the table above specifies the VPN as well:
>
> | VPN | PTE |
> |---|---|
> | 0 | 0xF5 |
> | 1 | 0xCA |
> | 2 | 0xFF |
> | 3 | 0x00 |
>
> For the data at virtual address 2048:
> 2048 == 100000000000 (in binary)
> First two bits, 10, indicate VPN 2.
> PTE for VPN 2 == 0xFF == 11111111 (in binary)
> Physical address == PFN + offset == 11 1100 0000 0000 (in binary) or 0x3C00
>
> For the instruction at virtual address 1024:
> 1024 == 010000000000 (in binary)

First two bits, 01, indicate VPN 1
PTE for VPN 1 == 0xCA == 1100 1010 (in binary)
Physical address == PFN + offset == 10 1000 0000 0000 (in binary) == 0x2800