Virtual Memory

Every program pretends that it has access to a 32-bit "address space".
But, the hardware and operating system know what's really going on and just maintain the illusion.

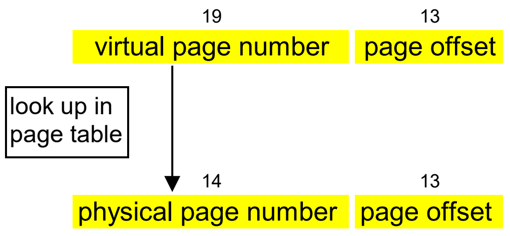
RAM is not 4 GB, only several megabytes. For our examples, we'll assume 128 MB.
Newer machines have even larger virtual memories, such as 42-bits which would be 4 TB.
(42-bit address held in a 64-bit register where 22 bits aren't used yet)

When fetching:

```
        if (reference is in cache - check tags)
                :)  cache hit --> one cycle
        else if (in RAM)
                :(  cache miss --> 10 to 100 cycles
        else
                !!!  page fault --> must go to disk, which could take ~ 1,000,000 cycles
                and the operating system will put our program on ice
```

Note that RAM typically holds (parts of) many running programs at once, so it will never be big enough…

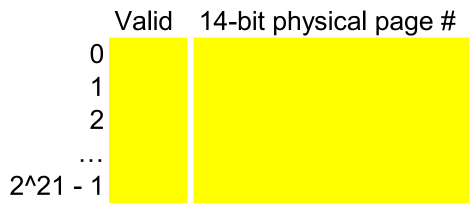
RAM acts like a "cache" for the disk.
The units of memory that we manage are called **pages**, rather than lines.
The size of a page is usually several kilobytes. Let's use 8 KB.



note: physical page = virtual page size

The page table will tell us if the page actually resides in RAM, or else we have a page fault.

The page table itself is in memory, so this is another reason why accessing RAM is slower than cache.



The size of this page table is about 2 * 2^21 bytes = 4 MB.

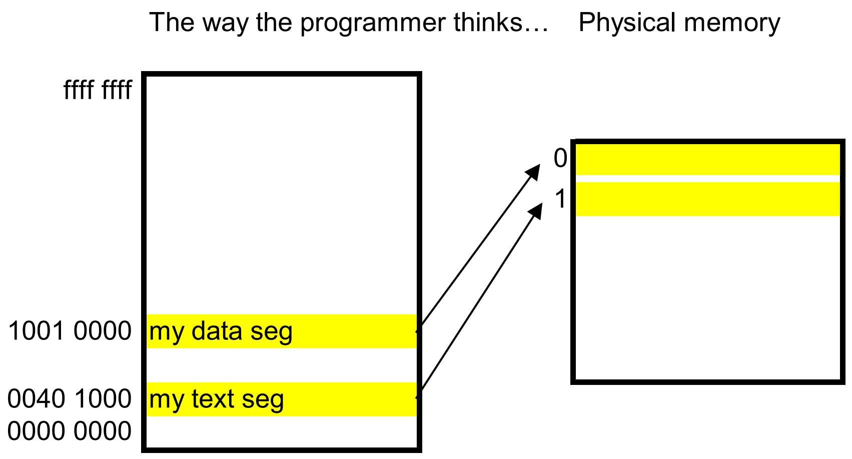
When a program starts, the OS loads the data segment and text segment from disk to RAM. For example, on the MIPS, we could start with:

one page of data	one page of text
0x1001 0000	0x0040 0000
0x1001 1ffff	0x0040 1ffff

A page would be sufficient to hold 2048 words (instructions).
For larger programs, we would need to load more than just one page to avoid future page faults.

helpful table of address sizes:

0x 10	16 bytes
0x 100	256 bytes
0x 1000	4 KB
0x 1 0000	64 KB
0x 10 0000	1 MB
0x 100 0000	16 MB
0x 1000 0000	256 MB
0x1 0000 0000	4 GB



Practice…

1. Let's break down the 32-bit addresses into virtual page number, and draw what the page table should look like when we load in the 2 pages above.

2. What are the virtual and physical addresses of the following:
    a. The 5th instruction in the program.
    b. Assuming the first thing we declare is a 100-element integer array, its 5th element.

3. Repeat the above steps using a memory system that has 256 MB of RAM and 4 KB pages.