Practice with memory system

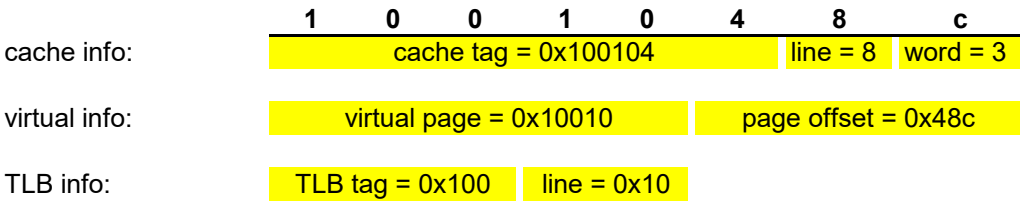
To keep our arithmetic simple, let's make these assumptions.

total virtual memory	4 GB
size of RAM	256 MB
size of cache	256 bytes
page size	4 KB
line size	16 bytes

Therefore, what can we say about the following:

1. Number of bits in a virtual address?
2. Number of bits in a physical address?
3. Number of virtual pages?
4. Number of physical pages?
5. Show how the virtual and physical addresses are partitioned.
6. How many lines are in the cache?
7. Show how the cache's tag, line number and word number are determined in the virtual address.

Let's see what happens when we reference something at this address: **0x1001048c**.



"Caching" is the communication between cache and RAM.
"Paging" is the communication between RAM and disk.

The **page table** tells us how to translate a virtual page number into a physical page number.
(the page offset will stay the same).

In our example, we're mapping from a 20-bit virtual page number to a 16-bit physical page number,
so note that the physical address will have 16+12 = 28 bits.

Because referencing the page table itself is rather slow (!), it has its own cache called the **TLB**.
The TLB is nothing fancy - just an abridged version of the page table.
It maintains info about our virtual pages, so notice we only need to partition part of address.

Let's say our TLB has 256 entries. For each entry, it stores a tag and a physical page number.

So, finally, our referencing algorithm looks like this:

```
if (cache tags match)                          cache hit
        :)
else if (TLB tags match)                       cache miss, but TLB hit
        load cache line from RAM
else if (in page table)                        cache miss, TLB miss, but hit in page table
        load cache line from RAM
        update TLB
else                                           cache miss, TLB miss, page fault!
        load cache line from RAM
        update TLB
        load page into RAM from disk
```