Virtual Memory on a 32 bit system.

The VM for a linux process on a 32 bit system is organized as shown below.

Each page has a virtual page number and a physical page number. The address mapping between the VM and Physical memory may look as shown below.
Each page is 4KB long. So, to address any location inside the page we need a 12 bit offset. Similarly, to locate a page anywhere in the memory we require a 20 bit address. So, effectively we require $2^{20}$ bits to keep track of all the pages of a process.

Page Table Organization:

The page table can be organized as vector table with $2^{20}$ entries each containing the entry for corresponding physical address of the page. This would require effectively $2^{20}$ bits of storage per process.

Alternately, we can have a bi-level page table hierarchy organized as shown below.
This is a preferred method over the previous one as it helps us save a lot of overhead memory. i.e; Some of the second level tables can be directly mapped to a page fault. Practically a given process only uses a very small amount of memory.

The only disadvantage is that the memory access is doubled, but it is still a very fair trade off.

**TLB: Translation Look aside Buffer:**

Each memory access in the architecture described above requires almost 2 more access to memory to get the Physical Page number. To improve over this design we add a caching module that saves the recent page mappings. This cache is called TLB.
Performance Issues:

It induces all the problems which would occur on adding a cache. It has to be cleared very often for keeping it integrity. This involves additional overhead to operations like context switching.