Updates:
PTE_U (idea from zarzees) can be very useful. It can be used to understand which page is for user (maybe?? Does code segment of a user process has PTE_U on?)

Physical page -> page in ram
NB:

- Keep a **refcount** var in a Swap struct, if you want to integrate COW
- Make new functions swapout, swapin ~~in swap.c~~ (where the lists are) to encapsulate everything related to the swaps
- All the structures can be kept in kalloc.c
- Or maybe preferably in separate file (I haven't done it yet)

Keeping track of live pages of user processes:

- Each live page is identified by **(procId, vpn(virtual page number))** pair.
- Whenever a **physical page is allocated** for a user process, the corresponding process id and vpn should be **stored in a list**.
- Physical page (Metadata) for a user process is mainly allocated in allocproc (for pagetable, trapframe and trampoline. These pages are probably not considered as user process pages. So I've ignored these pages.)
- **Pages alloced in *uvmalloc* are to be considered (send procID as param to uvmalloc) -> extra param in mappages (to track call from uvmalloc)**
- When a process ends, its related pages should be removed from live page list (done in unmappage (keep param for tracking where called from))

Swapping out
If number of user processes exceeds MAX_LIVE_PAGE,

- then take a entry out of the list, find its physical page address (from the page table or you can keep pte in the list),
- swap out the physical page,
- free it,
- mark the page table entry invalid (in the pagetables of all the processes using that page) and turn on the swapped bit ()
- keep track of the swap structure of the page.
- The swap structure could be saved in a list (required for swapping cow page) [works] or in the pte(the pointer of the swap structure could be stored in the upper bits of the page table entry) [riskyy]

Swapping in
When a swapped page is accessed, a page fault should occur. (I have found that sometimes it causes the same fault as COW). At that time, use the procId and vpn to find the swapped out structure (from the swapped list or from the pte), swap in, deal with reference count of swap(for cow)

1. Swap.c run, swap, and swapmem → page_info/
2. Uvmunmap and mappages new flag parameters
    a. kernel/user both use mappages-> kernel calls = flag 0

3. Uvmunmap and mappages new “pid” parameters
    a. Kernel call = pid 0 (dont care)
    b. User call = pid
       uvmcopy->mappages
       Add “pid” parameter to uvmcopy
       pid->uvmcopy->mappages