Inverted Paging:

⇒ In single-level paging, if 10 processes are running in CPU currently, then 10 different page table will be needed. Each page table for each page.

⇒ Currently running processes' page table must be in Main Memory.

⇒ So, we have to use Main Memory frames for storing many page tables.

⇒ Storing multiple page table (one PT for one Process) can lead to increased in Memory Usages. Thus, memory overflow can happen.

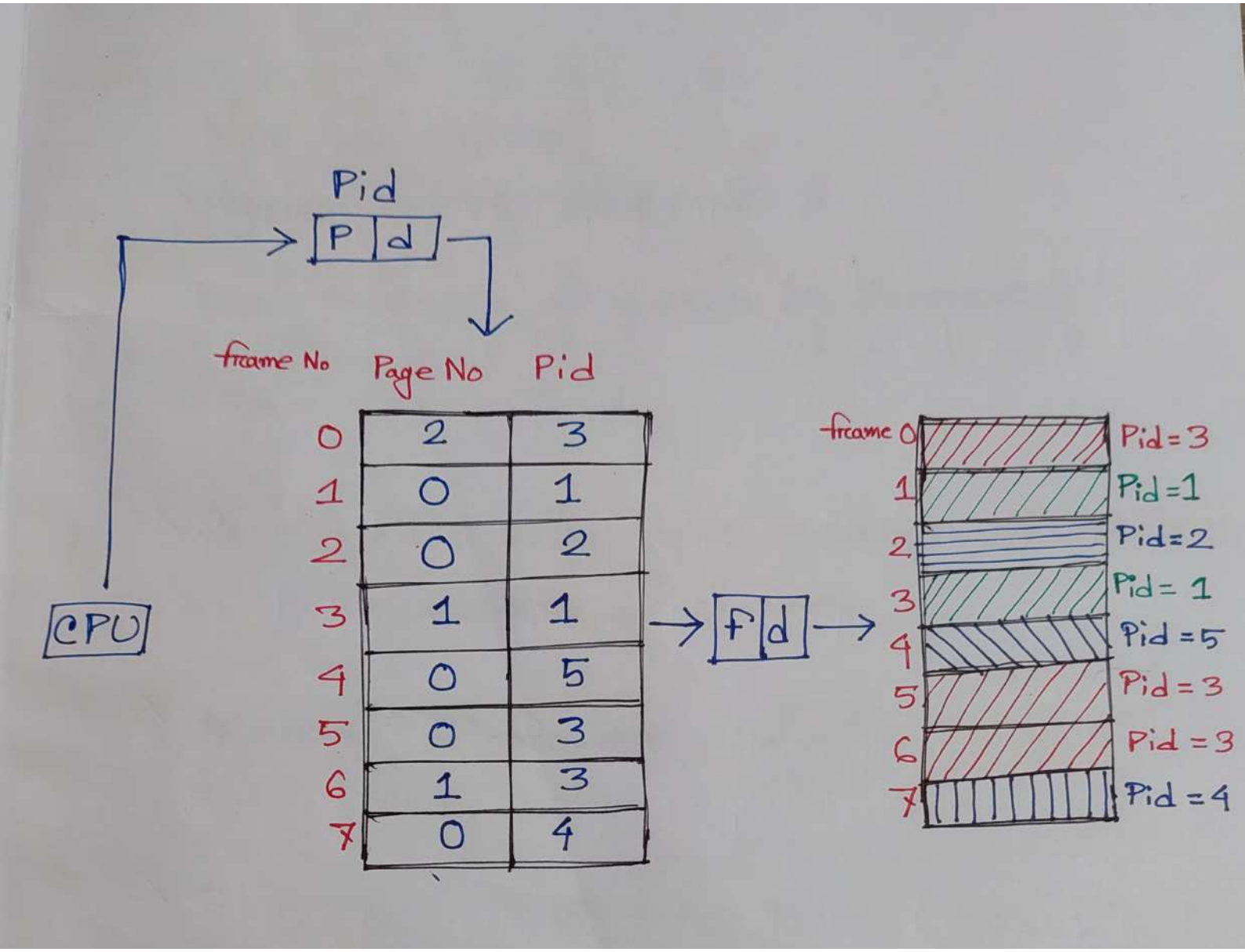
⇒ In inverted paging, a single global page table will be shared by all processes. Here, we do not use unique page table for each process.

- Number of Entries in Inverted Page Table = Num of Frames in M.M

- If we want to find frame number of logical address generated by CPU, then we have to perform linear search for finding corresponding process's page number.

- Linear Seach is inefficient in case of time complexity.

- So, inverted page table can reduce memory usage but also will lead to higher time complexity. So Inverted P.T is not popular.


Pid
P
d
CPU
frame No
Page No
Pid
0
1
2
3
4
5
6
7
2
3
0
1
0
2
1
1
0
5
0
3
1
3
0
4
f
d
frame 0
1
2
3
4
5
6
7
Pid = 3
Pid = 1
Pid = 2
Pid = 1
Pid = 5
Pid = 3
Pid = 3
Pid = 4

L.A = 64 bits

Page Size = 16 KB

Physical Memory Size = 8 GB

Find Number of entries in Inverted Page Table.

$\Rightarrow$

In inverted page table, number of entries = number of frames.

$$\text{Number of frames} = \frac{8\,GB}{16\,KB}$$

$$= \frac{2^3 \times 2^{30}}{2^4 \times 2^{10}}$$

$$= 2^{33-14}$$

$$= 2^{19}$$

So Num of entries $= 2^{19}$