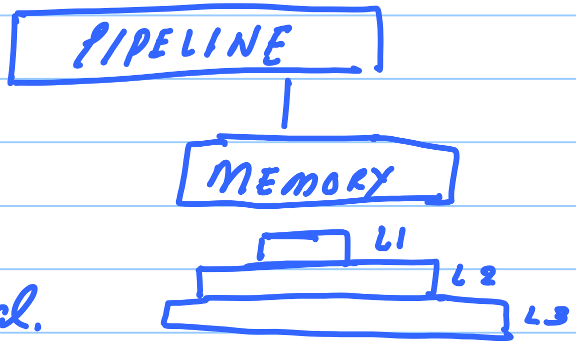
Note Title 31-10-2011

Oct - 31, Ropar.

Virtual Memory :



Example : WinEdt
Windows Journal.

Q: How do we ensure that one program is not overwriting the memory space of another program?

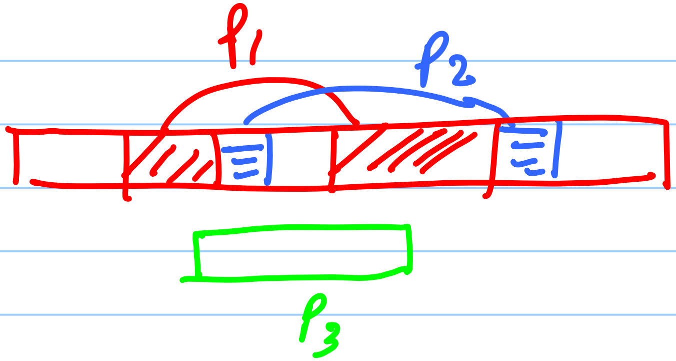
Ans: Programs are isolated from each other.

$P1 \quad P2$

assign disjoint memory regions to P_1 & P_2

How & when do I assign?

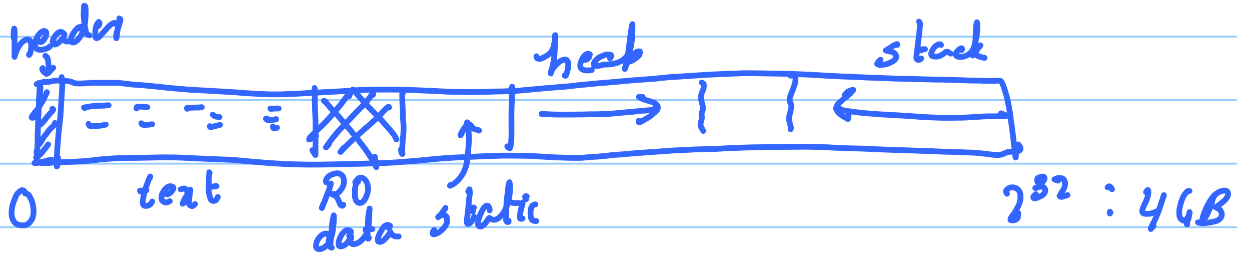
* Statically (compile time)
  - → cannot eliminate overlaps.
  - → not possible to assign large contiguous memory regions.



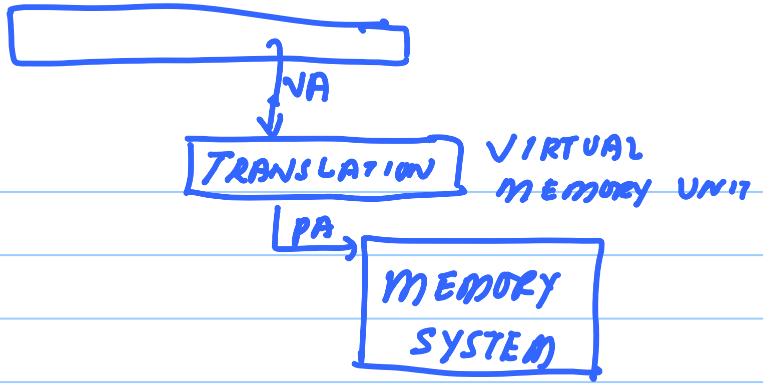
32-bit address space

Virtual Memory: Every program assumes that it has $\underbrace{\text{4 GB}}_{\text{VA space}}$ of memory exclusively for itself.

virtual address space.



If all programs have the same layout, how come they don't conflict?



VA : Virtual Address

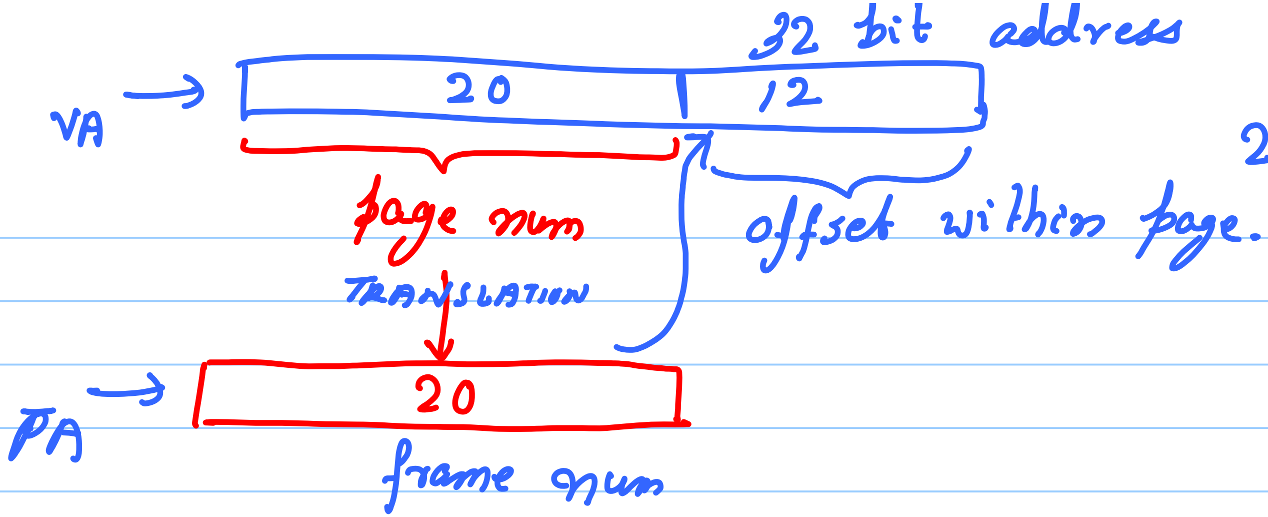
PA : Physical Address.

Translation

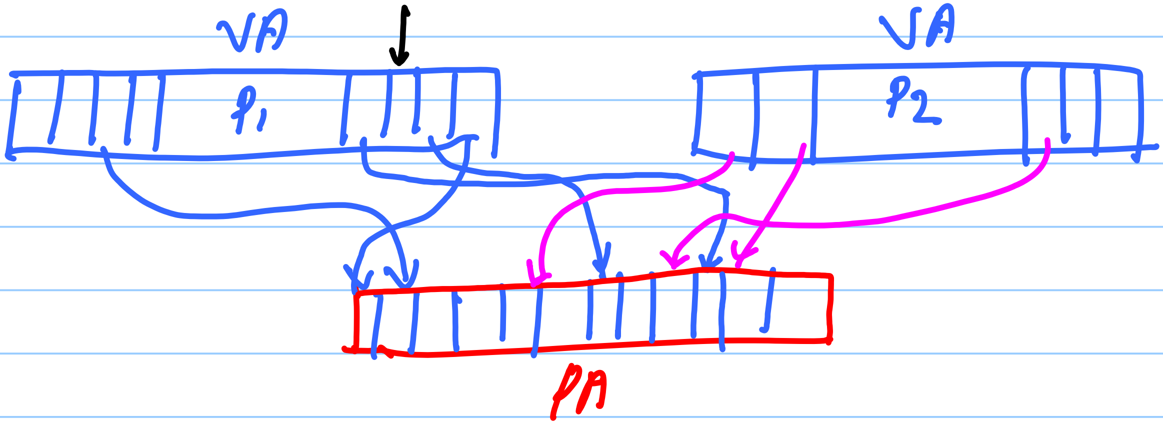
Break the virtual address space into chunks of 4 kB (page)

Break physical address space into chunks of 4 kB (frame)

Page Size = Frame Size.

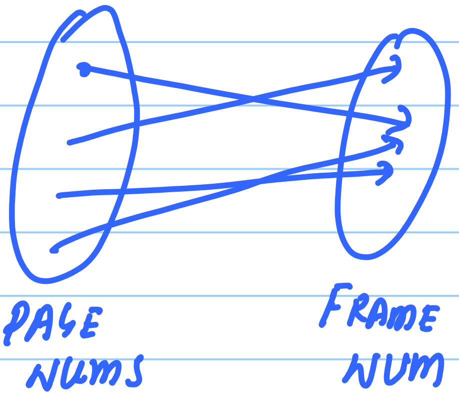

32 bit address
VA
20
12
page num
offset within page.
TRANSLATION
PA
20
frame num

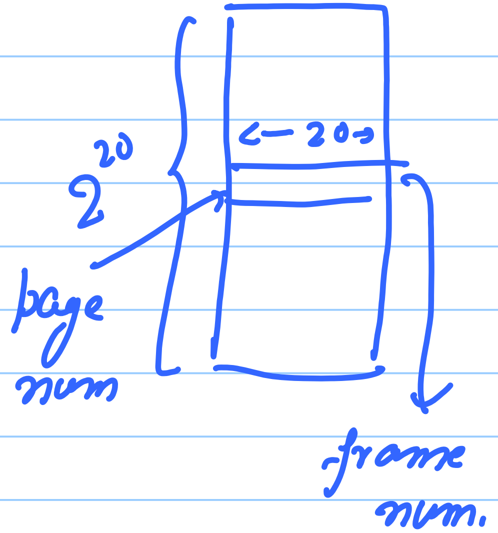
$2^{12} = 4kB$


VA
p_1
VA
p_2
PA

For each program:

Mapping Table → Page Table





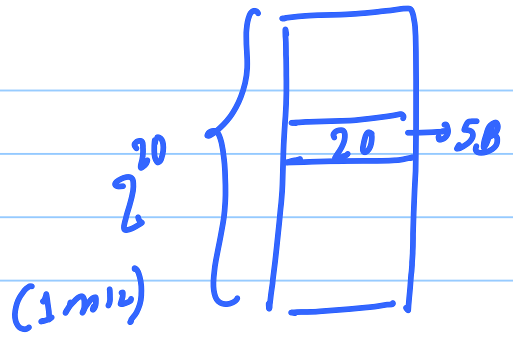
Page Table → .

By maintaining mappings.

1) Isolation between programs.

2) Notion of VA space

Single Level Page Table



(5MB) size → ① large size

② very slow.

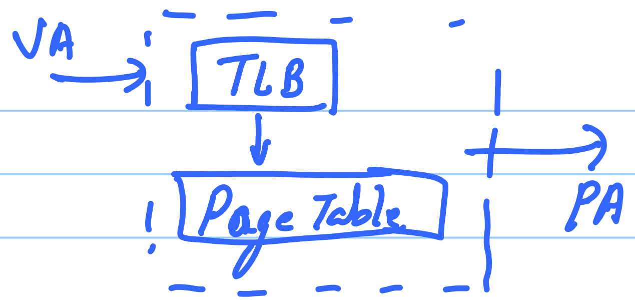
very slow

TLB → Translation Lookaside Buffer

32 or 64 (n) most recently used mappings in a fully-associative cache.

Most accesses have a TLB hit

VM



Next Class: Multi-Level Page Tables