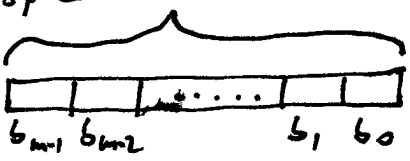
Radix Sort

not comparison based
assumes array elements are b-bit integers

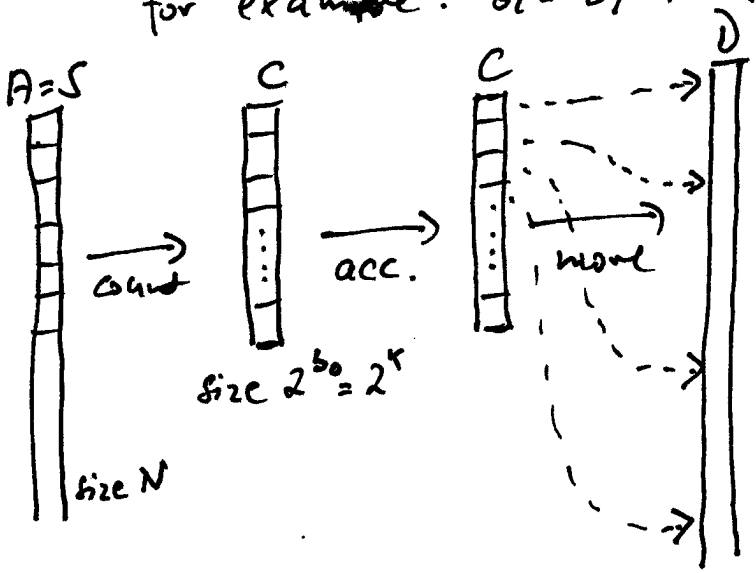
Basic idea: slides

steps:
- count (histogram)
- accumulate
- move

In general: split b bits into chunks



for example: $b_i = b/m = k$ (constant radix k)



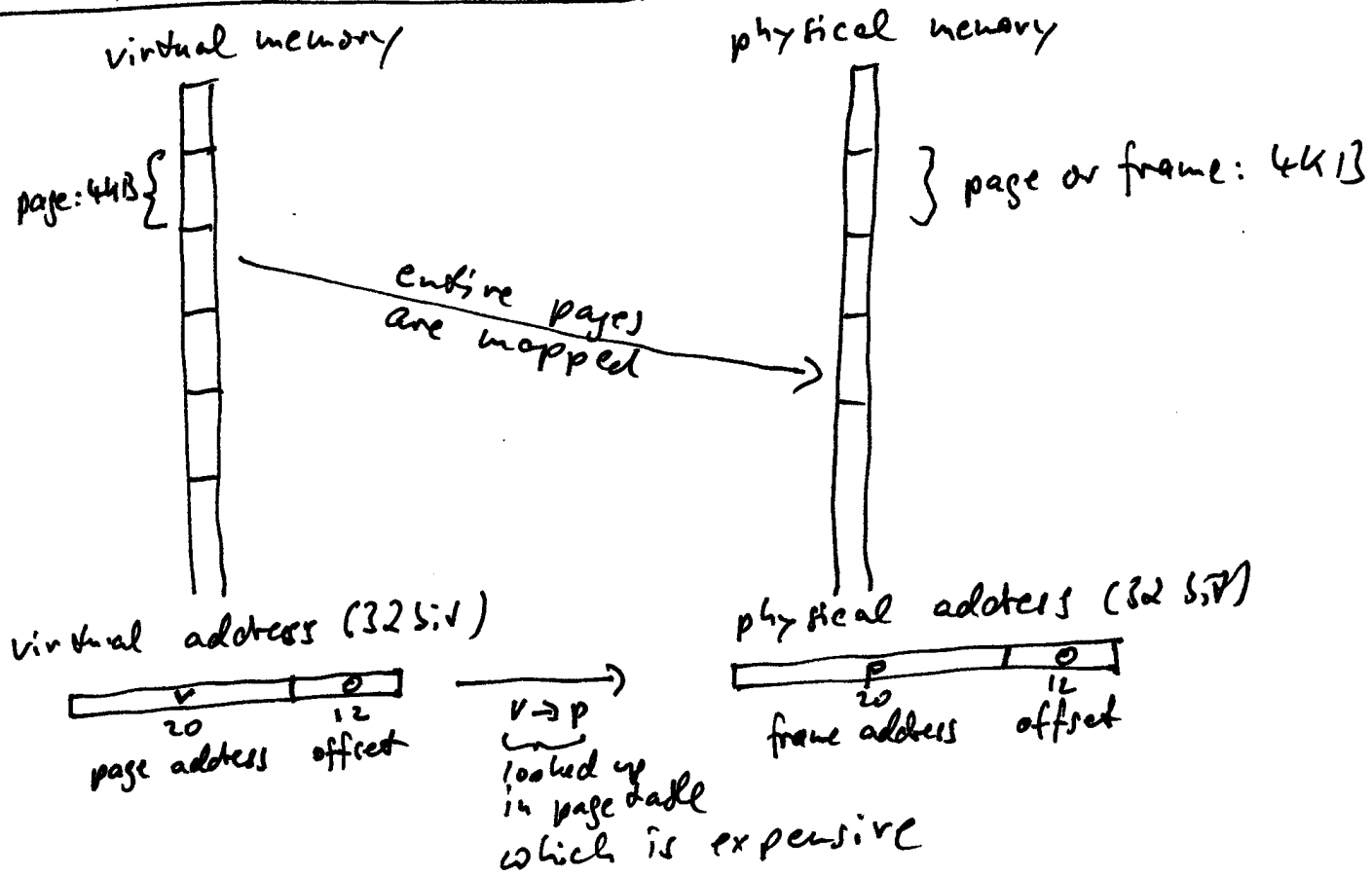
writes consecutively but at 2^{b_0} different locations

Analysis:

count: $O(N)$
acc.: ~~$O(k)$~~ $O(2^k)$
move: $O(N)$
m iterations: $O(mN)$

locality: temporal: no
spatial: yes, but 2^k "live" address regions
$\Rightarrow$ can cause TLB misses

Translation Lookaside Buffer (TLB)



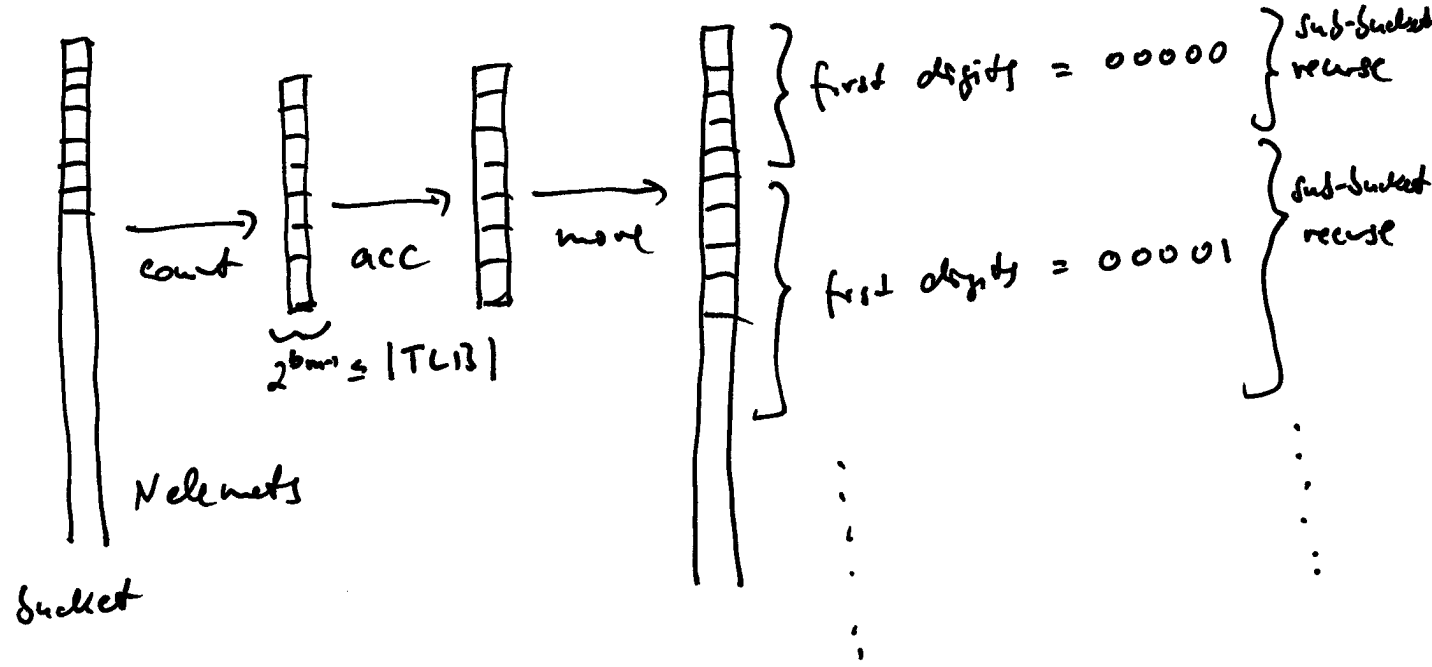
TLB: stores (caches) recent (vp, p) pairs

Intuitively: The size of the TLB determines how many different memory regions a program can operate on simultaneously and efficiently.

Core 2 Duo:
- L1 virtually addressed
- L2 physically addressed
- $|TLB| = 32 = 2^5$

CC-Radix Sort (CC = Cache Conscious)

Basic idea:
- Sort by MSB first → temporal locality
- Choose radix k s.t. $2^k \leq |TLB|$
- repeat until arrays fit into L2 cache
  then use standard radix sort



Adaptive Sorting

slides