Page Mode – DRAM

-To access a byte one needs to access the whole page.
-High Price paid initially in terms of time taken to get the first byte , but the successive ones are pretty quick.

Memory Interleaving – Used to compensate for the slow speed of DRAM
– Memory Bandwidth for commodity DRAM is around 6400 MB / s.

Pipelined Design is used in building router hardware .
– Every cycle of execution (just like in a architecture pipeline) we complete a stage . In essence we are completing 1 task every cycle. (Ignore the stalls due to memory access)
– when such operations are being formed for the task , we make use of Wide Word Parallelism . For example , a 64 bit processor allows one to do 64 bit wise operations in 1 cycle .
– Always try to take advantage of the full length of the processor.

IP Prefix Lookups and Exact Match Look up (chp 10 from book)

An example of a switch in which there is a table of addresses . When a packet / frame comes along we need to decide which output port should we forward this frame to .

IP addresses is an example application where Prefix Look ups are needed as opposed to MAC addresses where a complete / Exact Match is needed .

Solutions proposed for the above :

1) Hash Table (software solution )
1. When we have a very good hash functions , the number of collisions are few and we can still attain an O(1) lookup on average , but even then with a large number of IP addresses the collision can increase and we end up chaining them on a single bucket , thus increasing latency.
2. Need another solution.
2) Perfect Hashing
1. Given a STATIC set of keys to be hashed into the hashtable , we can attain perfect hashing. Perfect hashing lets us attain the latency of O(1) .
2. The keys here is analogous to MAC addresses.
3. The important thing to notice here is the the key set is static , meaning the set of addresses is constant over a long period of time and does not change.
4. The first step towards hashing function is to find a good hash function that hashes only one key into each bucket. For this , suppose we have a family H of hash functions ( not enumerated ) we pick up a bunch of hash functions and test if the condition is met. If yes , we have found a perfect hash function for our constant set of keys , else we repeat this process . So eventually one ends up sampling the H space for the best hash function.
We propose such kind of a solution since Look ups are the most frequent operations as opposed to add() or delete() which happen probably once / day or once / week .

3) CAM – Content Addressable Memory.