Computer Architecture: Fundamentals, Tradeoffs, Challenges

Chapter 8: Cache Memory

Yale Patt

The University of Texas at Austin

Austin, Texas

Spring, 2023

Outline

- ***What is it? Why do it? How?***
- ***Examples***
- ***The Abstraction***
- ***Characteristics***
- ***Some additional comments***
- ***Cache Coherence***

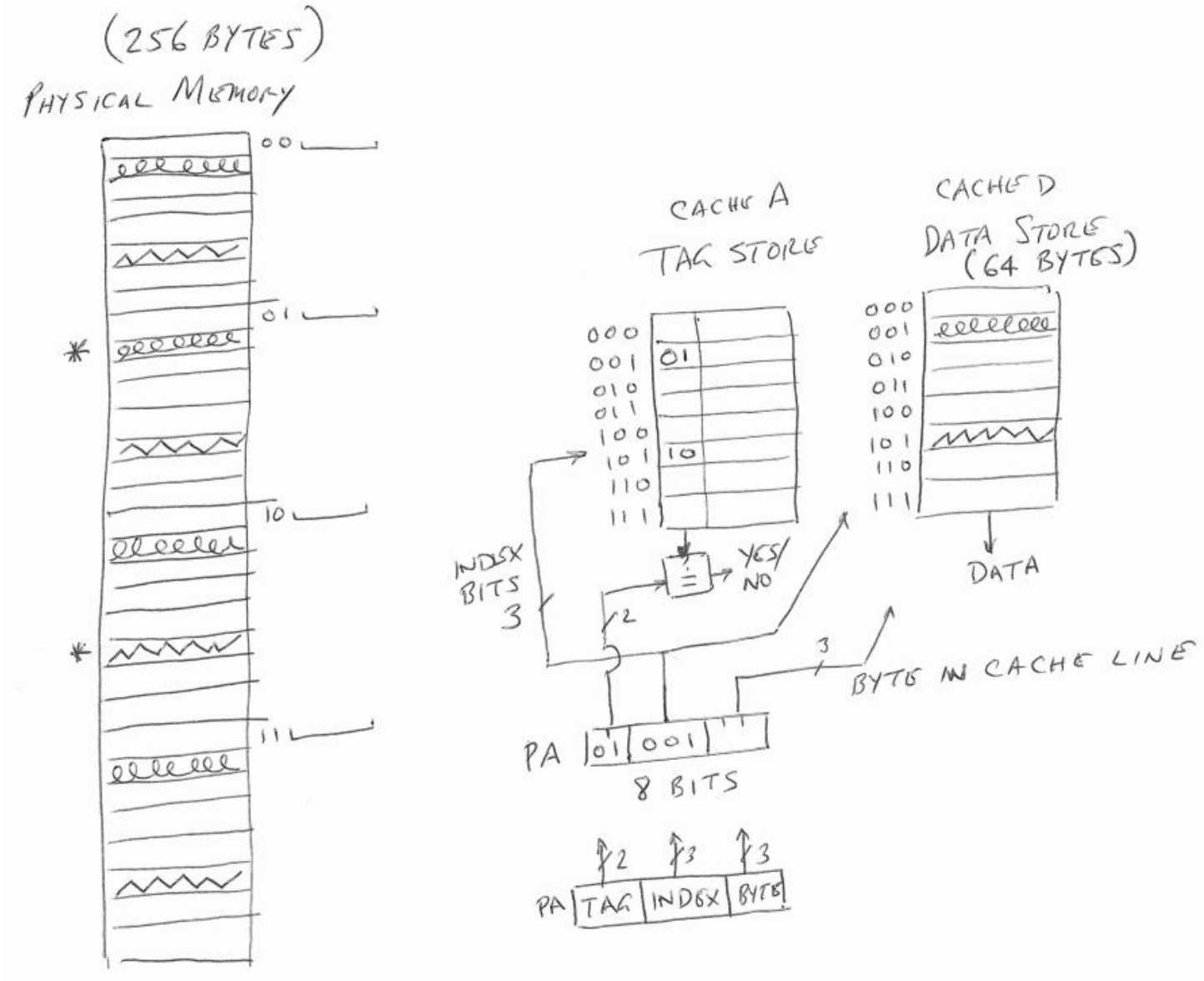
Cache Memory

- ***What is it?***
  - ***A small piece of memory that is on the chip***
- ***Why do it?***
  - ***Cache access takes x time units, Memory takes 100x units***
  - ***Temporal locality: you probably will want to access again***
  - ***Spatial locality: you probably will want items close by***
- ***How?***
  - ***When a miss occurs, bring on-chip a larger unit of memory***
  - ***Granularity is a line size (aka block size) [Spatial locality]***

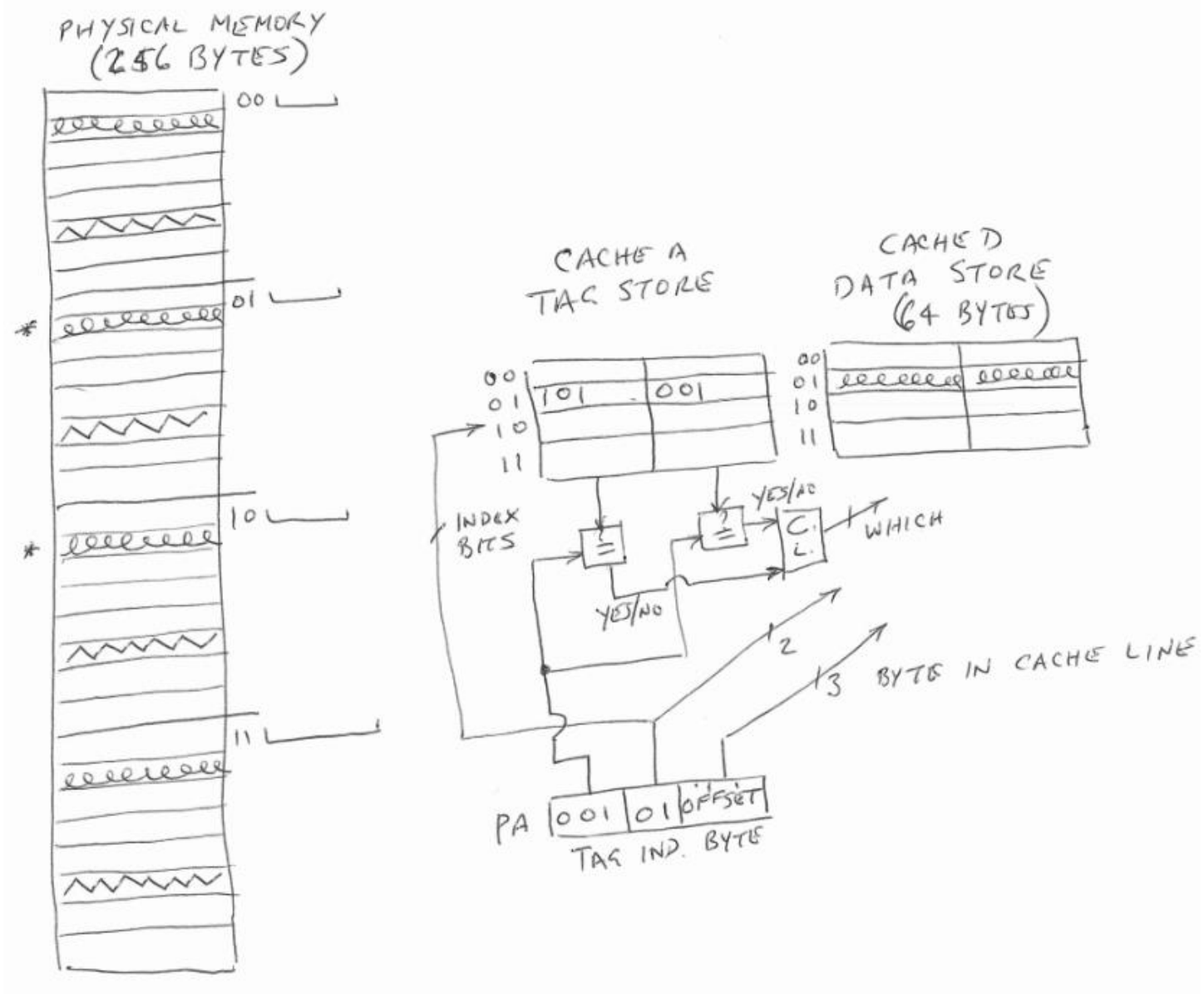
Two Examples

- ***Memory:** 256 bytes, consisting of 32 lines, 8 bytes per line*
- ***Cache***
  - ***Two parts***
    - ***Data Store (contains the stored instructions/data, identifies the size of cache)***
    - ***Tag Store (one entry per data store line, used for bookeeping information)***
  - ***Size of cache: 64 bytes***
  - ***8 byte line size***
  - ***8 lines in the cache***
- ***Example 1***
  - ***Direct mapped (4 lines in memory compete for one cache way)***
  - ***8 sets, each having one way (2^index bits = number of sets)***
- ***Example 2***
  - ***2-way set associative (8 lines in memory compete for two ways***
  - ***4 sets, each having two ways (2^index bits = number of sets)***

An example: 64 bytes, 8 byte line, direct mapped



An example: 64 bytes, 8 byte line, 2 way set assoc.



The Abstraction (Physically addressed cache)

Virtual address

V

TLB (to get the PFN)

V

Physical address

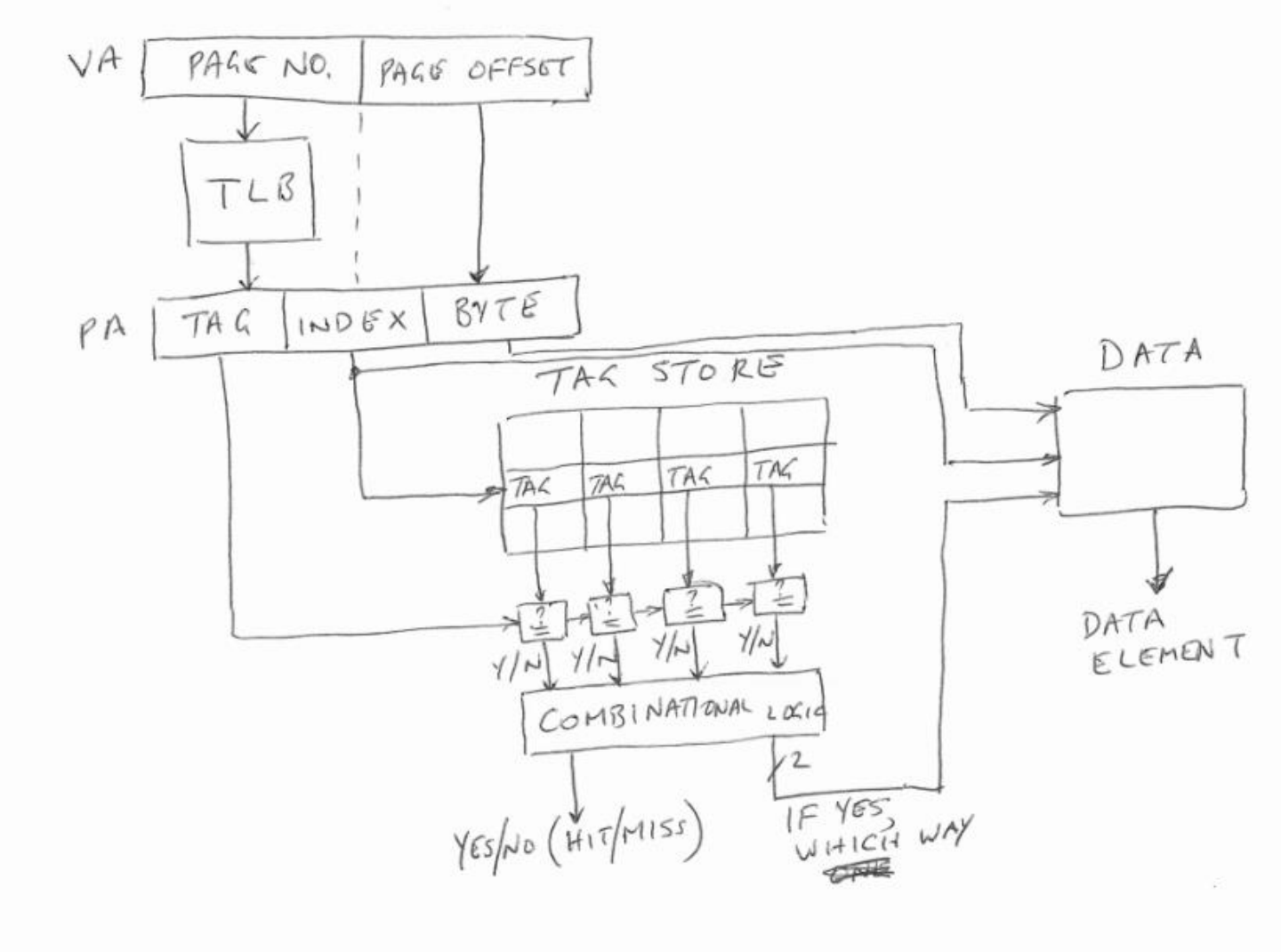
V

Tag Store (to see if and where it is)

V

Data Store (to get it)

The Access

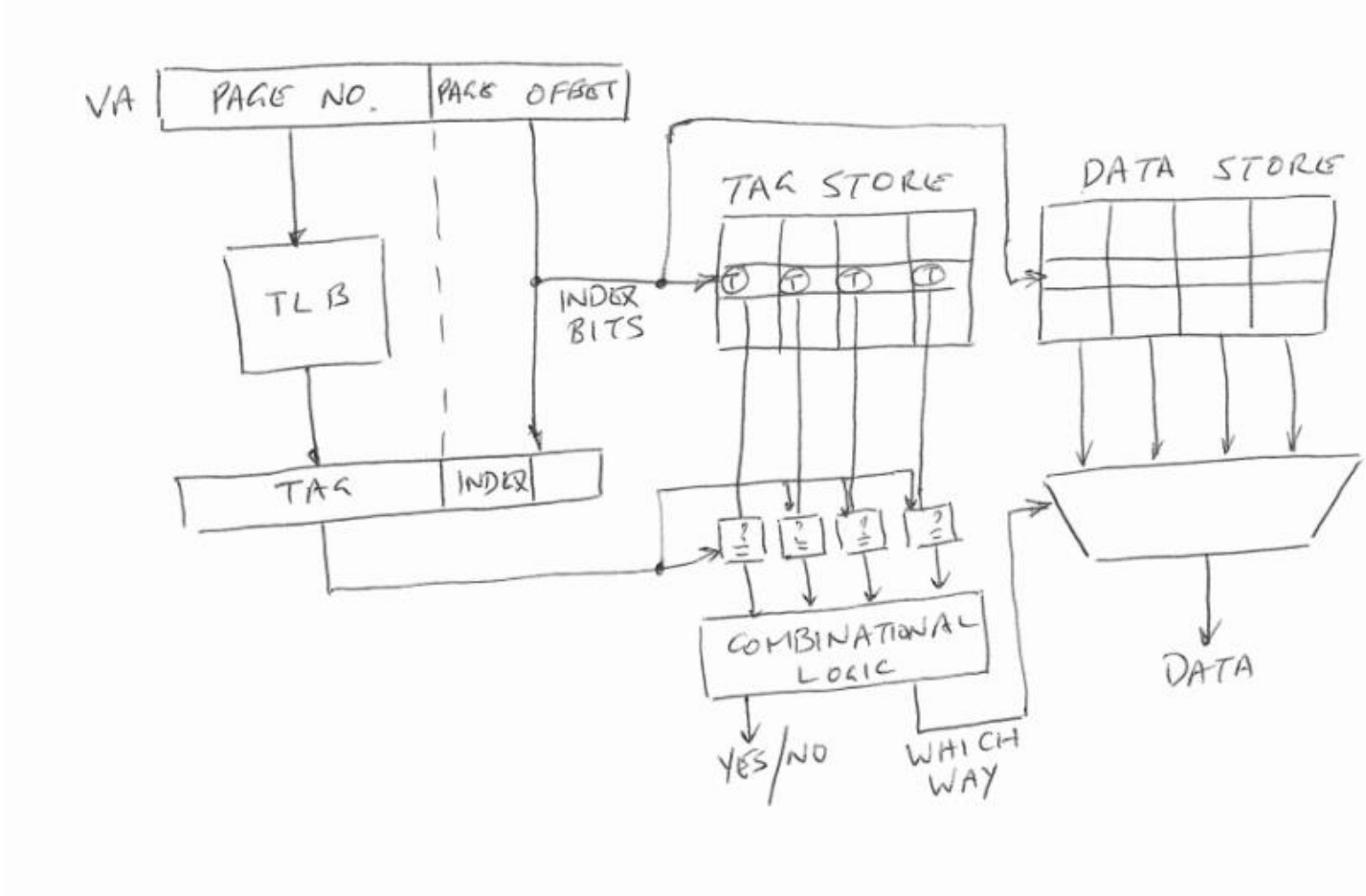


Three Sequential Accesses! Can we do better?

Can we do better? Answer: Yes (Both Physical and Virtual)

- ***All physical: index bits in unmapped part of address***
- ***All virtual***
- ***Virtually indexed, physically tagged***

All Physical (unmapped index bits, output MUX)

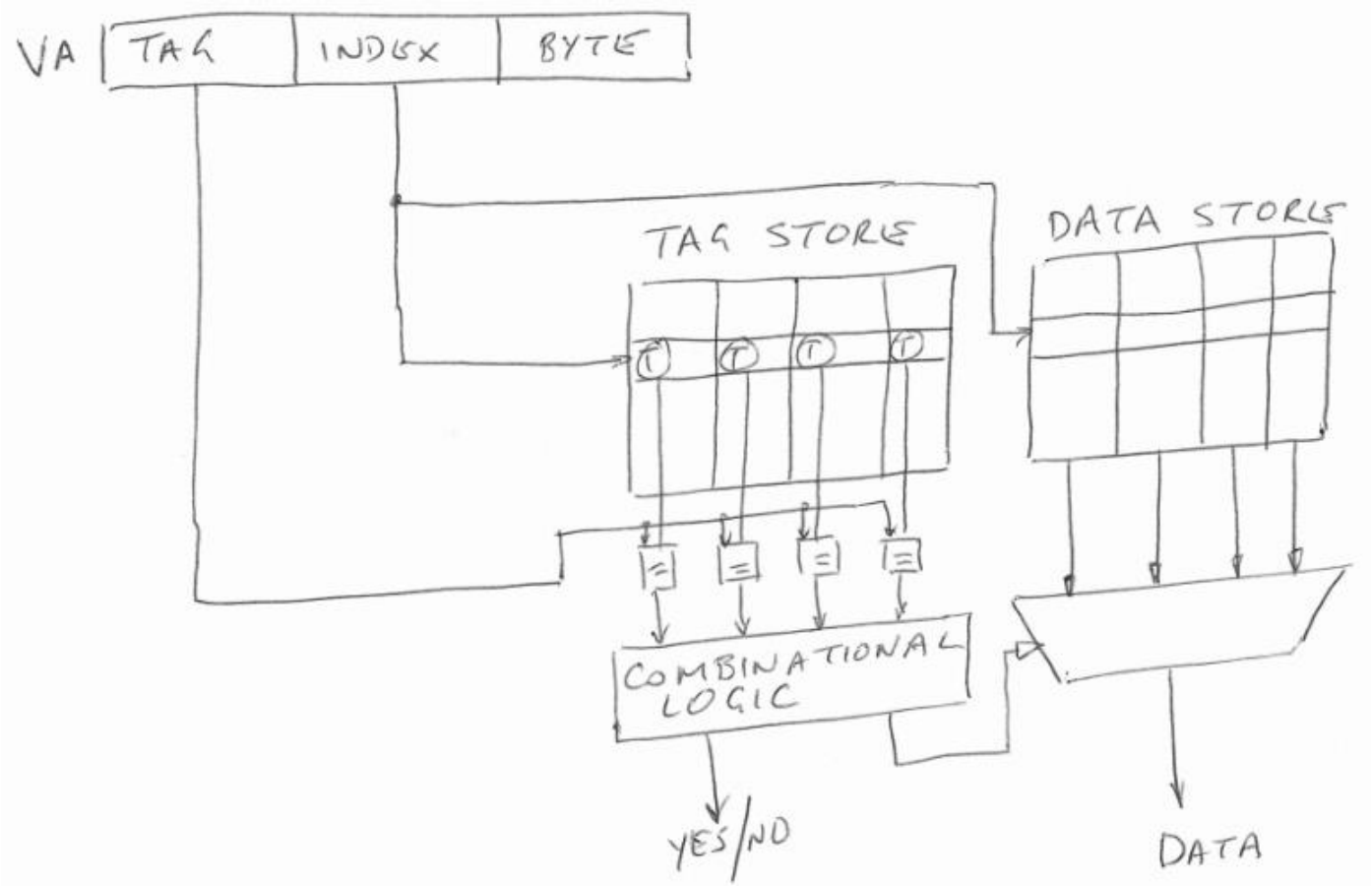


We can access TLB, Tag Store, and Data concurrently

Minus: Limits Size of the cache,

(although we can increase associativity)

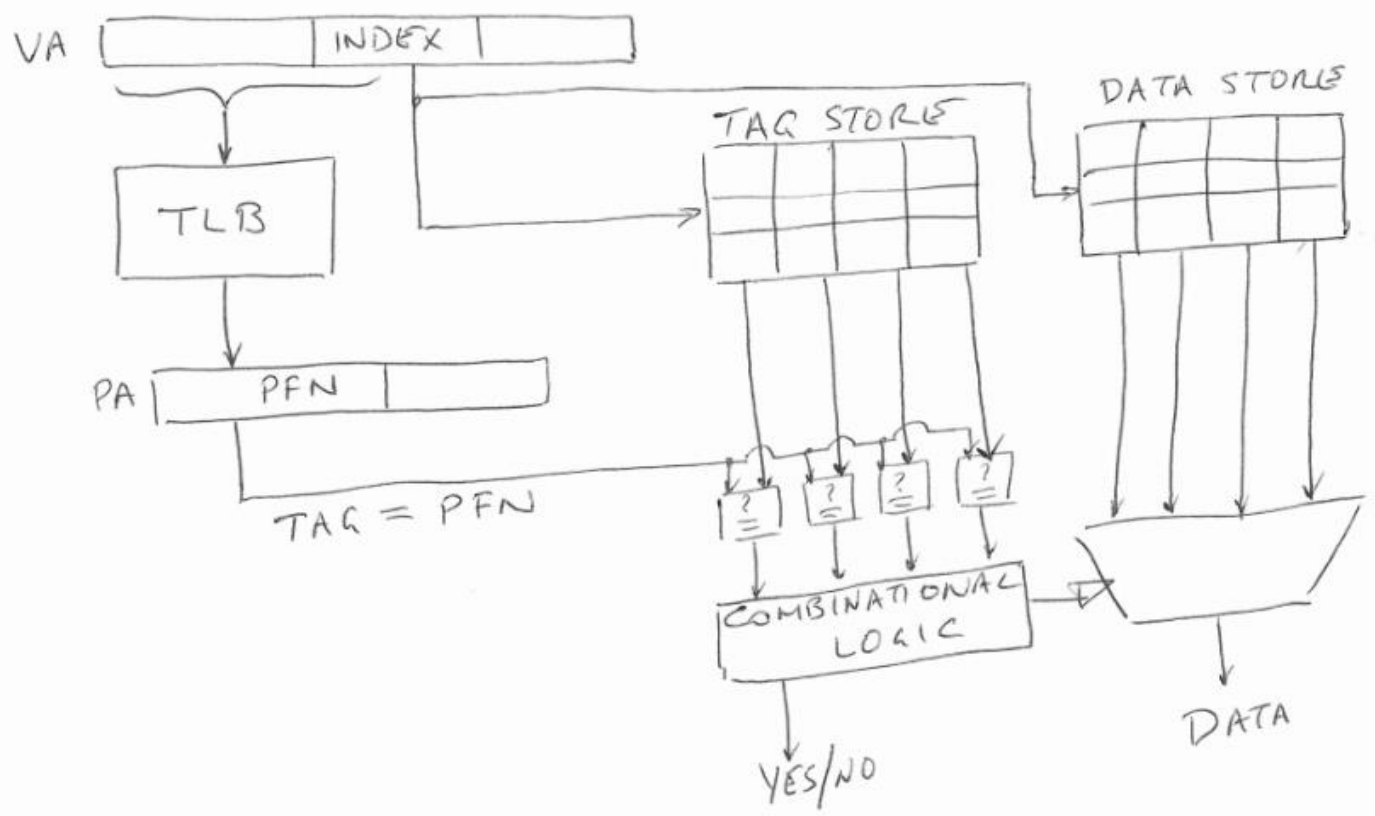
All Virtual



No TLB access is necessary (only using VA)

Flush on context switch (or, include process id)

Virtually indexed, physically tagged



TLB and Tag Store can be accessed concurrently

Minus: One PA may be in two places (Synonym Problem)

Characteristics

- ***Set associative***
  - ***Fully associative***
  - ***Direct mapped***
  - ***Affect on cache hit ratio, hits/(hits + misses)***
- ***Write back, Write through***
- ***Replacement policy***
- ***Separate unified instructions/data***
- ***Separate supervisor/user***
- ***Virtual/Physical***

Write through vs Write back

- ***Definition***
  - ***Write through: write to the cache, also write to memory***
  - ***Write back: write to cache does not also write to memory***
    - ***Write to memory happens when cache line is removed from the cache***

- ***Issues***
  - ***Simplicity of design***
  - ***Bus traffic***
  - ***Application environment (e.g., stack frame)***
  - ***Allocate vs non-allocate on a write miss***

Replacement policy

- ***Policies: LRU, FIFO, Random***
- ***Two-way set associative (a single bit per set)***
- ***Four-way (pseudo-LRU)***
  - ***One solution: 3 bits per set***
    - ***One bit for ways A,B, one bit for ways C,D, one bit for AB, CD***
  - ***Another solution: 8 bits, 2 bits for each way***
    - ***Victim bit: least recently used, will be evicted on cache miss***
    - ***Next victim bit: will become victim when victim is evicted***
    - ***Example (way B is least recently accessed, D is next least):***

	Victim	Next Victim		Victim	Next Victim
Way A	0	0		0	1
Way B	1	0	Acess to way B	0	0
Way C	0	0	>	0	0
Way D	0	1		1	0

A Tag Store Entry

- ***One per cache line in the Data Store (for bookkeeping)***
- ***Tag bits (to verify the memory address)***
- ***Valid bit (so you know if you can believe the tag bits)***
- ***Dirty bit (for write back caches: memory is obsolete)***
- ***LRU bit(s)***
  - ***used to implement the replacement policy***
  - ***Sometimes per entry, sometimes per set***

Some Additional Comments

- ***First cache: Motorola 68020***
  - ***Mid-1980s, one level, 256 bytes, instruction cache***
- ***Today, several levels***
  - ***Cache hierarchy (to approximate large storage, fast access)***
  - ***Level 1: Separate caches for Instructions and Data***
    - ***Pipeline requires concurrent access to instructions and data***
  - ***Level 2: Unified instruction/data***
    - ***Concurrent access is much less often***
  - ***Level 3: Because we can!***
- ***Inclusion property (everything in L1 is in L2***
  - ***Important for coherence***
- ***Prefetch***
  - ***Under software control, cache is part of ISA***
  - ***By microarchitecture, not visible to software, not part of ISA***

Cache Coherence

- ***Mostly important when we study multiprocessors***
  - ***Cache Coherence (by hardware; what is/isn't guaranteed)***
    - ***Same address in two caches MUST contain the same value***
    - ***No guarantees as to what the value is***
  - ***Memory Consistency (by software, needs cache coherence)***
    - ***If you care what the value is***
    - ***Needs cache coherence to make it work***
- ***Even in a uniprocessor***
  - ***IF we have intelligent controller***
  - ***e.g., DMA loads memory, independent of the processor***
    - ***Processor sees one thing in cache, Memory sees something else***

Grazie!