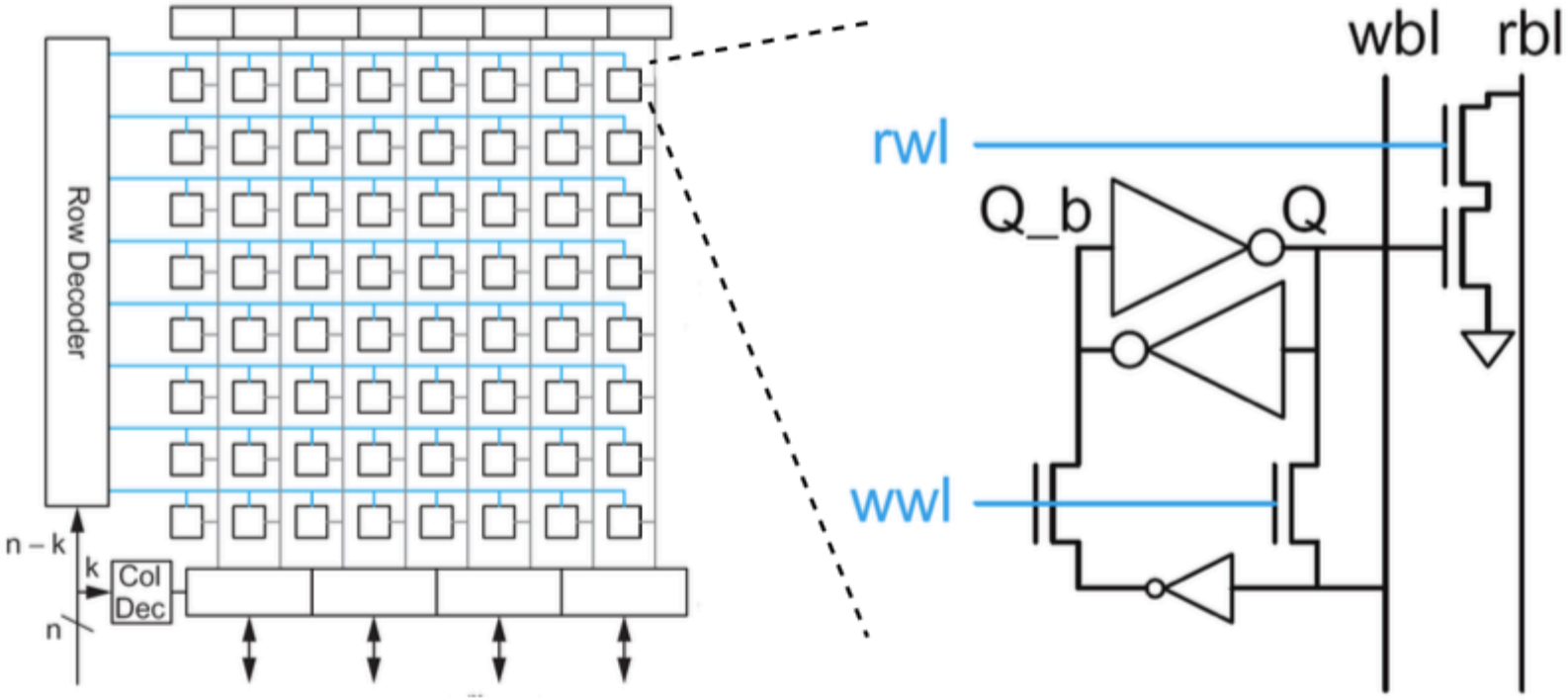
Lecture 14
Fundamental Memory Concepts (Part 2)

Xuan ‘Silvia’ Zhang
Washington University in St. Louis

http://classes.engineering.wustl.edu/ese566/

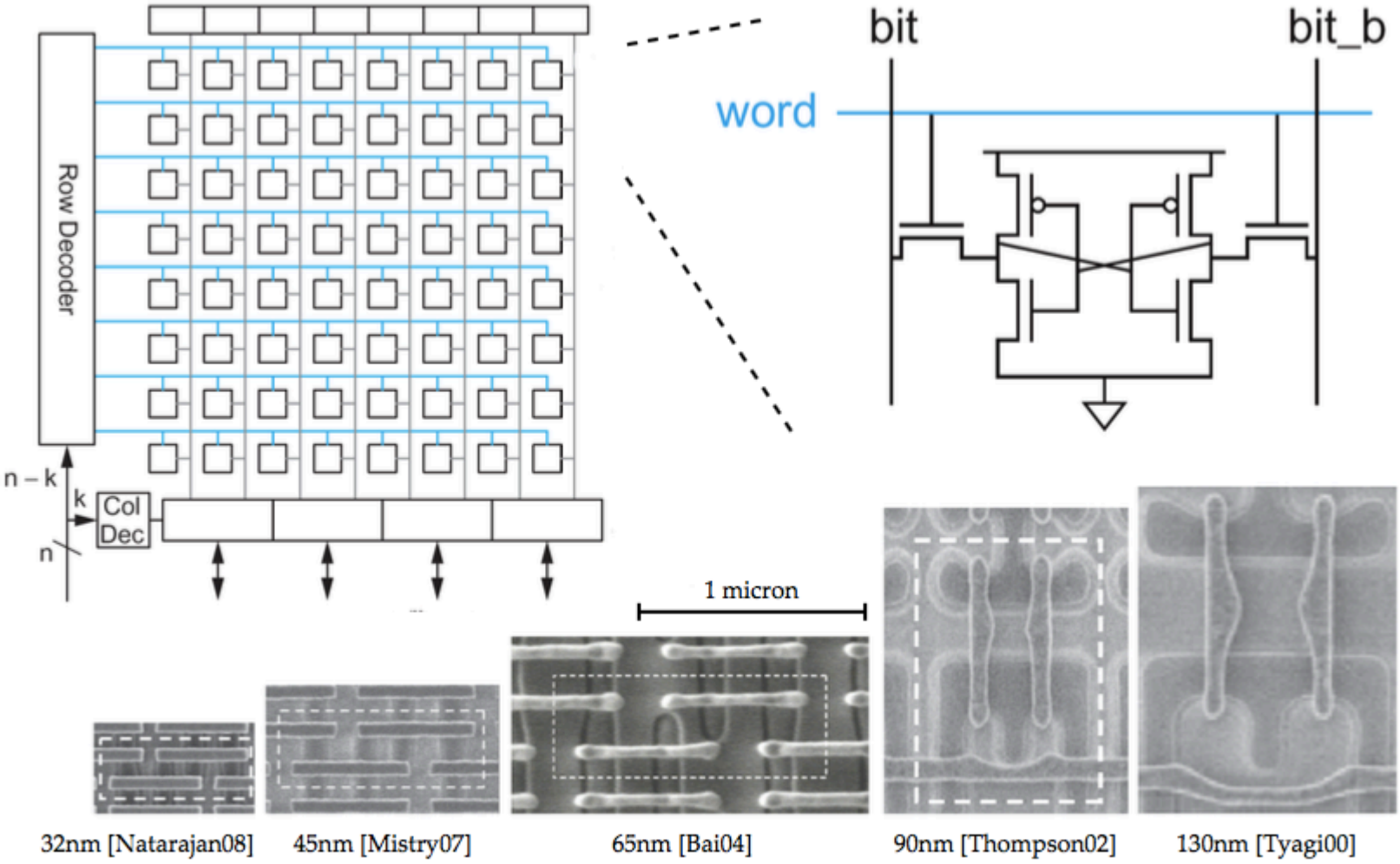
Memory Structure and Technology

- Register Files



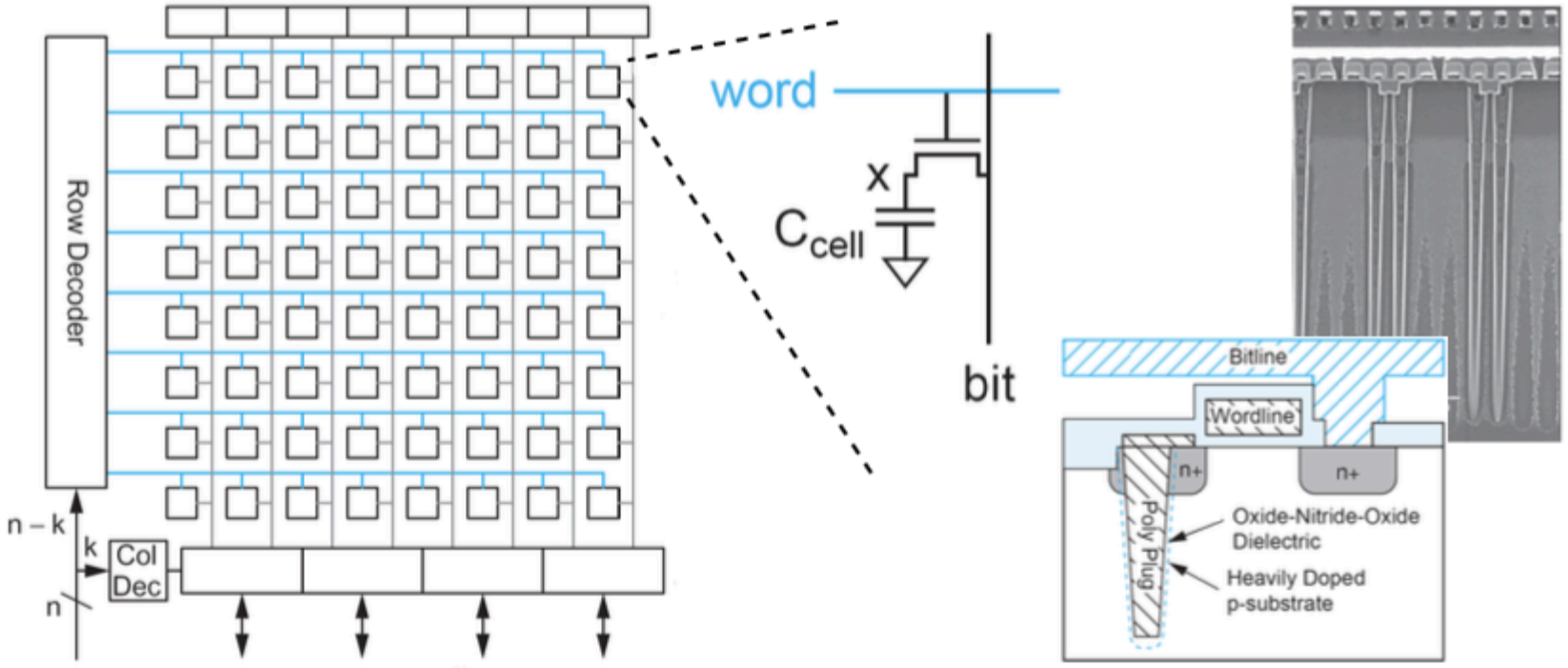
Memory Structure and Technology

- SRAM (cache, on-chip)



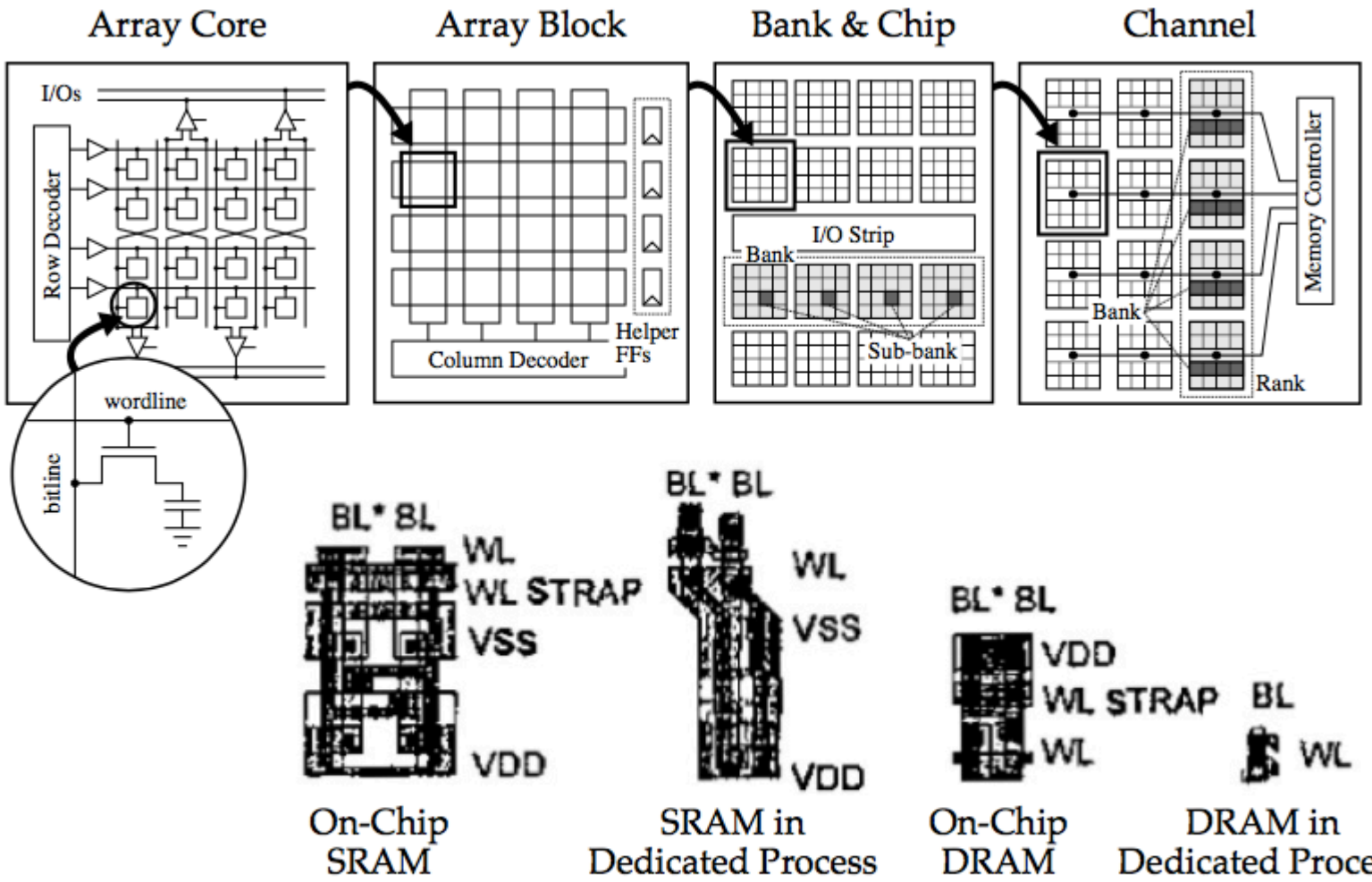
Memory Structure and Technology

- DRAM



Memory Structure and Technology

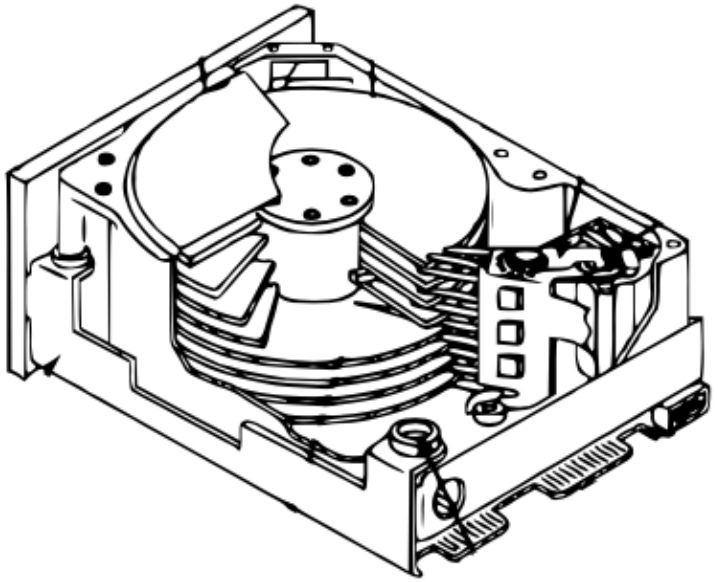
- DRAM



Adapted from [Foss, "Implementing Application-Specific Memory." ISSCC'96]

Memory Structure and Technology

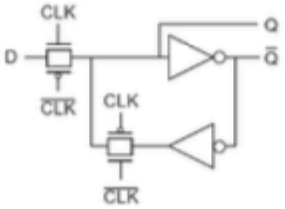
- Disk
  - magnetic hard drives require rotating platters resulting in long random access times with have hardly improved over several decades



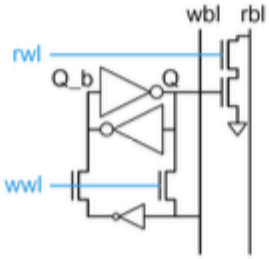
- Flash
  - solid-state drives using flash have 100x lower latencies, but also lower density and higher cost

Memory Technology Trade-offs

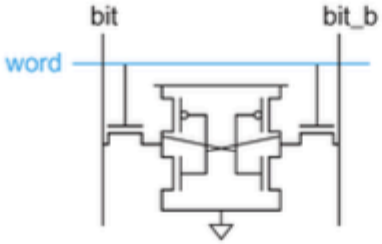
Latches & Registers



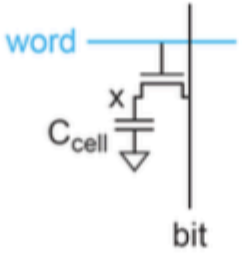
Register Files



SRAM



DRAM



Flash & Disk

Low Capacity
Low Latency
High Bandwidth
(more and wider ports)

High Capacity
High Latency
Low Bandwidth

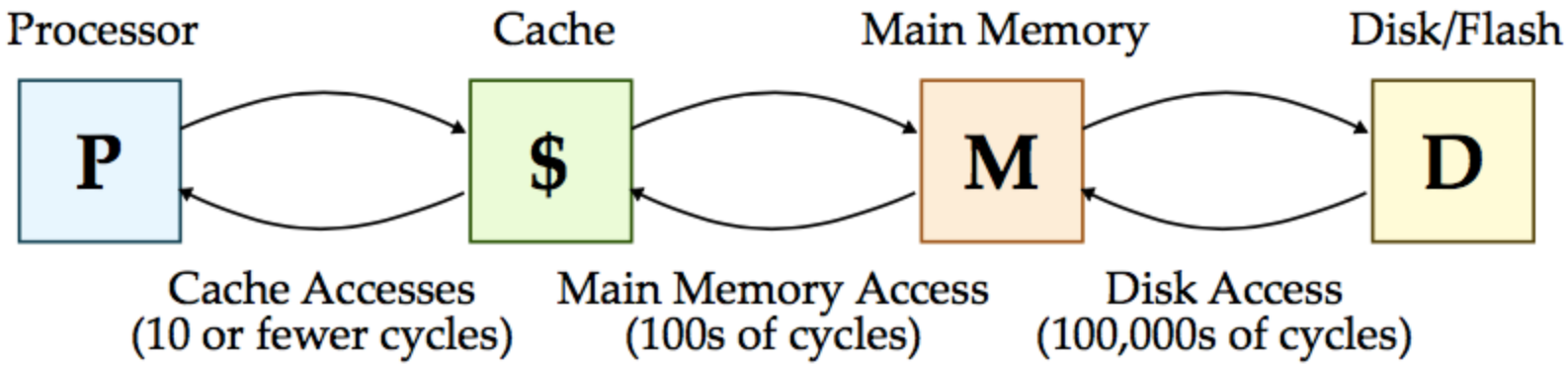
Latency Numbers: every programmers (architect) should know

L1 cache reference	1 ns
Branch mispredict	3 ns
L2 cache reference	4 ns
Mutex lock/unlock	17 ns
Main memory reference	100 ns
Send 2KB over commodity network	250 ns
Compress 1KB with zip	2 us
Read 1MB sequentially from main memory	9 us
SSD random read	16 us
Read 1MB sequentially from SSD	156 us
Round trip in datacenter	500 us
Read 1MB sequentially from disk	2 ms
Disk random read	4 ms
Packet roundtrip from CA to Netherlands	150 ms

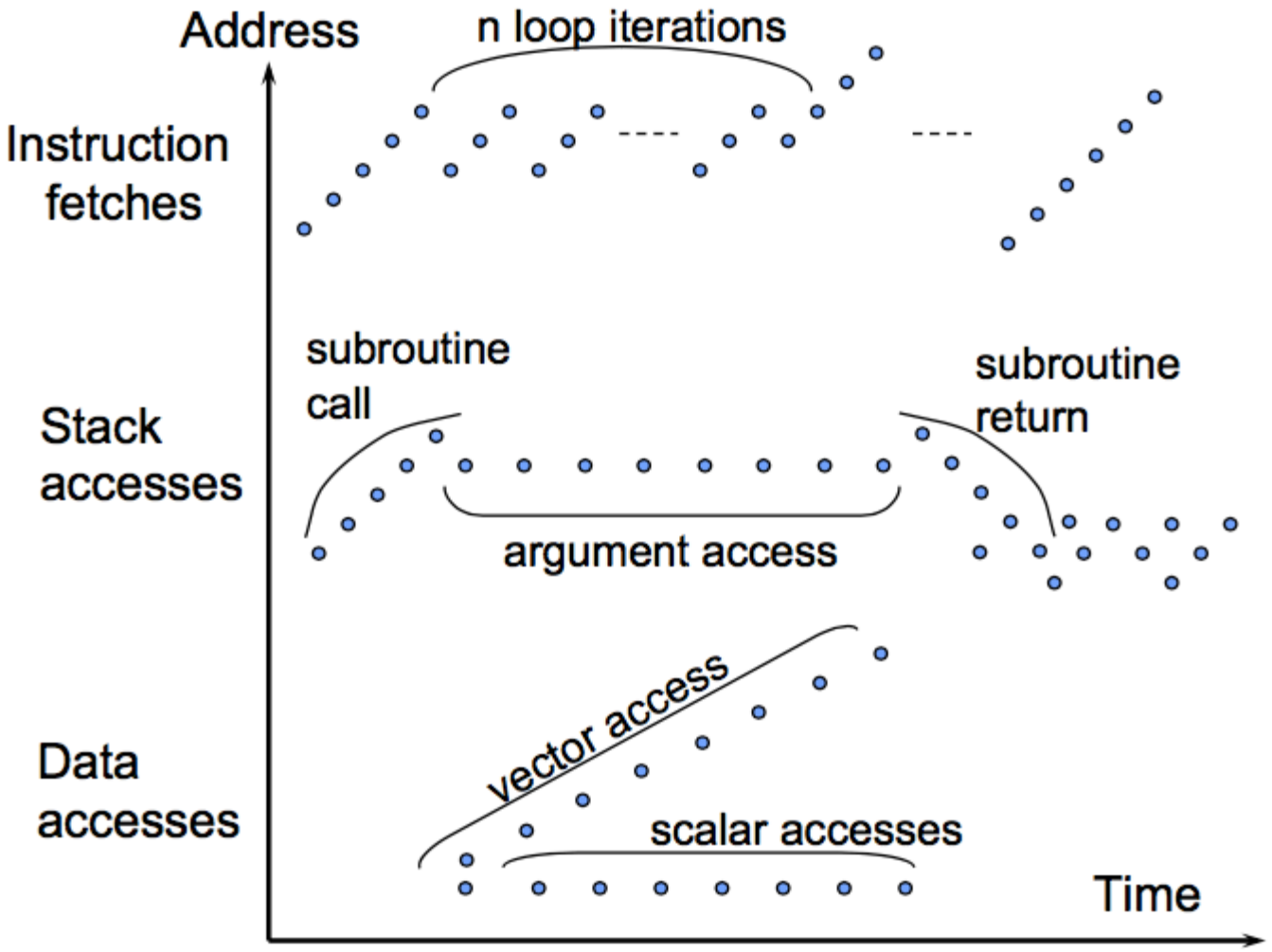
find updated at https://people.eecs.berkeley.edu/~rcs/research/interactive_latency.html

Cache Memories in Computer Architecture

- Three key questions
  - how much data is aggregated in a cache line
  - how do we organize multiple lines in cache
  - what data is replaced to make room for new data when cache is full
- Categorizing misses
- Write policies
- Multi-level cache



Typical Access Pattern
instruction vs data access, temporal vs spatial locality



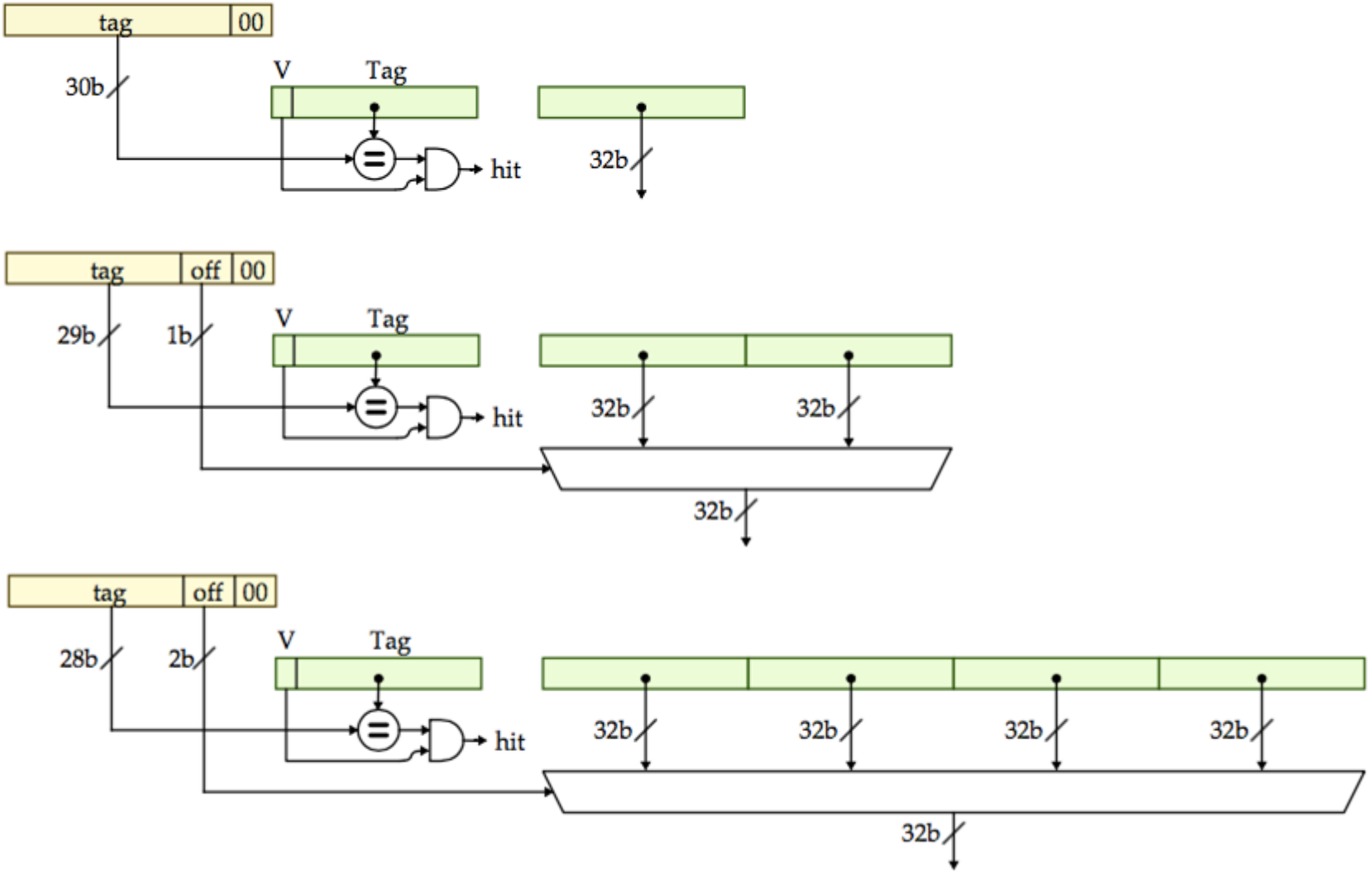
Understand Locality

Examine each of the following assembly programs and rank each program based on the level of temporal and spatial locality in both the instruction and data address stream on a scale from 0 to 5 with 0 being no locality and 5 being very significant locality.

	Inst Temp	Inst Spat	Data Temp	Data Spat

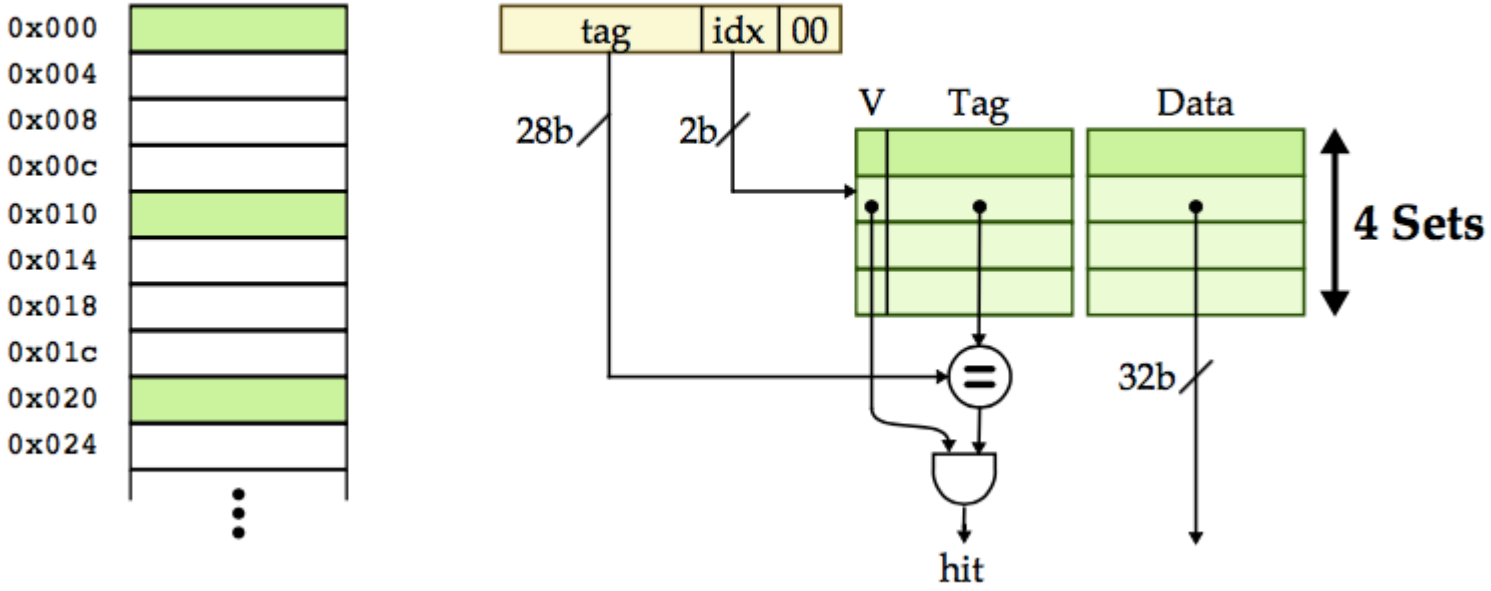
```
loop:
  lw    r1, 0(r2)
  lw    r3, 0(r4)
  addiu r5, r1, r3
  sw    r5, 0(r6)
  addiu r2, r2, 4
  addiu r4, r4, 4
  addiu r6, r6, 4
  addiu r7, r7, -1
  bne   r7, r0, loop
```

Q1: How Much Data is Stored in a Cache Line?



Q2: How to Organize Multiple Lines in the Cache?

Four-line direct-mapped cache with 4B cache lines



Example execution worksheet and table for direct-mapped cache

Dynamic Transaction Stream

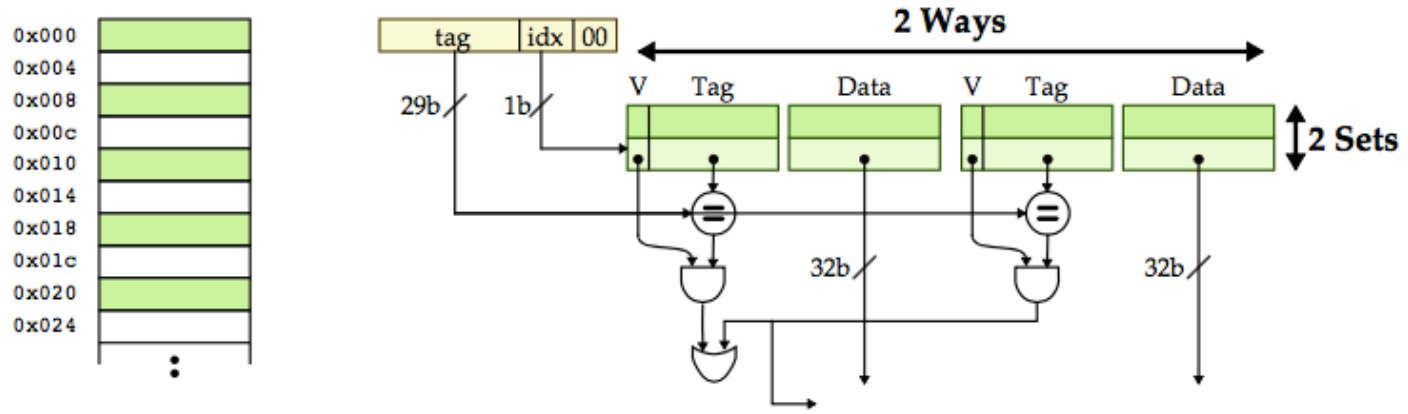
```
rd 0x000
rd 0x004
rd 0x010
rd 0x000
rd 0x004
```

0x000	13
0x004	14
0x008	15
0x00c	16
0x010	17

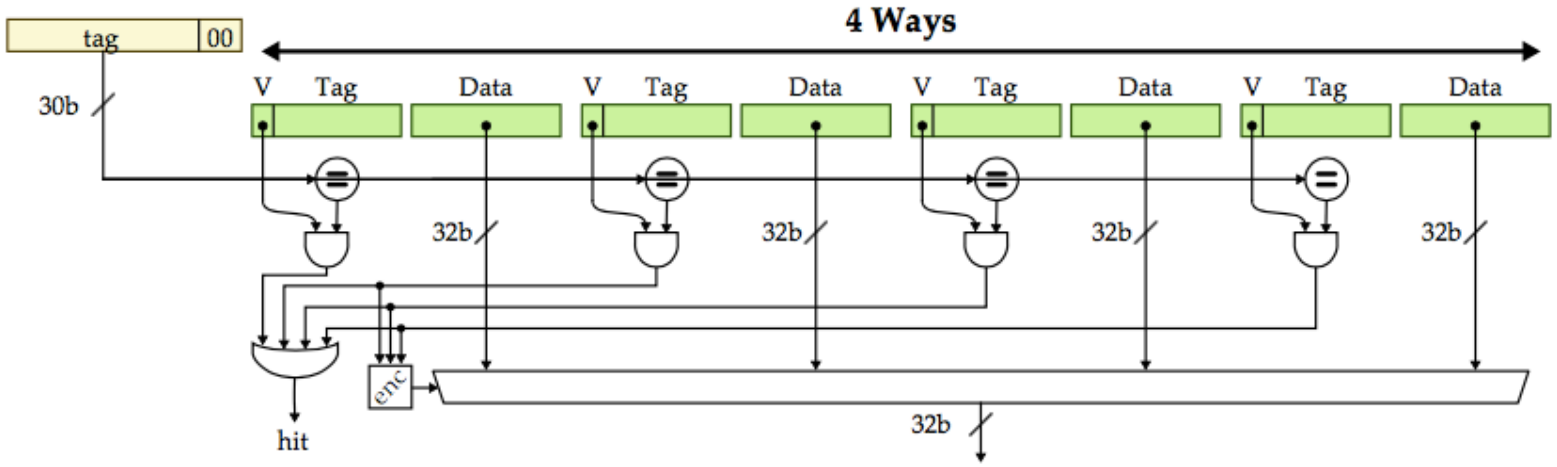
	V	Tag	Data
Set 0			
Set 1			
Set 2			
Set 3			

Increase Cache Associativity

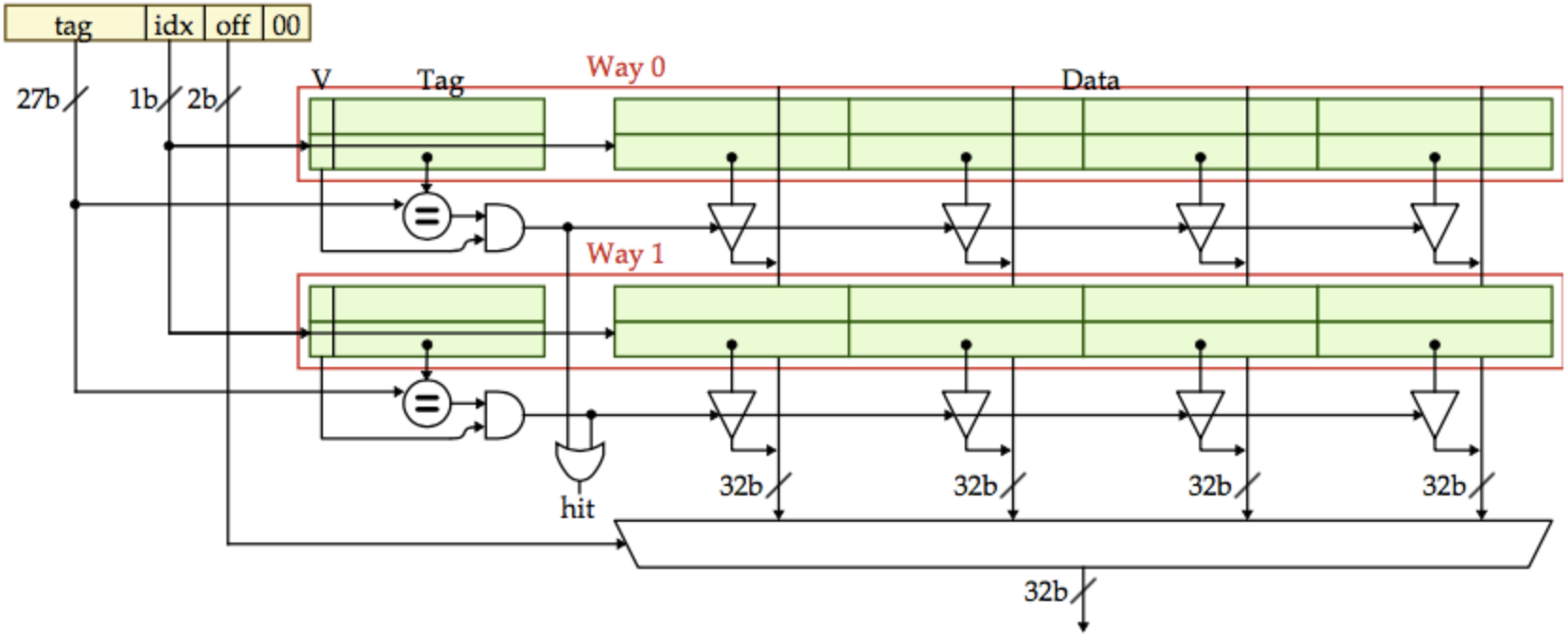
Four-line two-way set-associative cache with 4B cache lines



Four-line fully-associative cache with 4B cache lines



Increase Cache Line Size



Q3: What Data to be Replaced When Cache is Full?

- No choice in a direct-mapped cache
- Random
  - good average case performance, but difficult to implement
- Least recently used (LRU)
  - replace cache line which has not been accessed recently
  - exploits temporal locality
  - LRU cache state must be updated on every access
  - two-way cache can use a single “last used bit”
  - pseudo-LRU uses binary tree to approximate LRU for higher associativity
- First-In First-Out (FIFO, Round Robin)
  - simpler implementation without exploiting temporal locality
  - potentially useful in large fully-associative caches

Categorize Misses: The Three C's

- Compulsory
  - first-reference to a block
- Capacity
  - cache is too small to hold all of the data
- Conflict
  - collisions in a specific set

Classify Misses as a Sequence of Three Questions

- Q1: would this miss occur in a cache with infinite capacity?
  - if “yes”, then this is a compulsory miss
  - if “no”, consider Q2
- Q2: would this miss occur in a fully-associate cache with the desired capacity?
  - if “yes”, then this is a capacity miss
  - if “no”, then consider Q3
- Q3: would this miss occur in a cache with the desired capacity and associativity?
  - if “yes”, then this is a conflict miss
  - if “no”, then this is not a miss—it is a hit!

Examples

Assume we have a direct-mapped cache with two 16B lines, each with four 4B words for a total cache capacity of 32B. We will need four-bits for the offset, one bit for the index, and the remaining bits for the tag.

	tag	idx	h/m	type	Set 0	Set 1
`rd 0x000`						
`rd 0x020`						
`rd 0x000`						
`rd 0x020`						

Assume we have a direct-mapped cache with two 16B lines, each with four 4B words for a total cache capacity of 32B. We will need four-bits for the offset, one bit for the index, and the remaining bits for the tag.

	tag	idx	h/m	type	Set 0	Set 1
`rd 0x000`						
`rd 0x020`						
`rd 0x030`						
`rd 0x000`						

Write Policies

- Write-through with no write allocate
  - on write miss, write memory but do not bring line into cache
  - on write hit, write both cache and memory
  - require more memory bandwidth, but simpler implementation
- Write-back with write allocate
  - on write miss, bring cache line into cache then write
  - on write hit, only write cache, do not write memory
  - only update memory when a dirty cache line is evicted
  - more efficient, but more complicated to implement

Translation, Protection, and Virtualization

- Translation
  - mapping of virtual address to physical address
- Protection
  - permission to access address in memory
- Virtualization
  - transparent extension of memory space using disk

Most modern systems provide support for all three functions with a single paged-based memory management unit (MMU).

Analyze Memory Performance

$$\frac{\text{Time}}{\text{Mem Access Sequence}} = \frac{\text{Mem Accesses}}{\text{Sequence}} \times \frac{\text{Avg Cycles}}{\text{Mem Access}} \times \frac{\text{Time}}{\text{Cycle}}$$

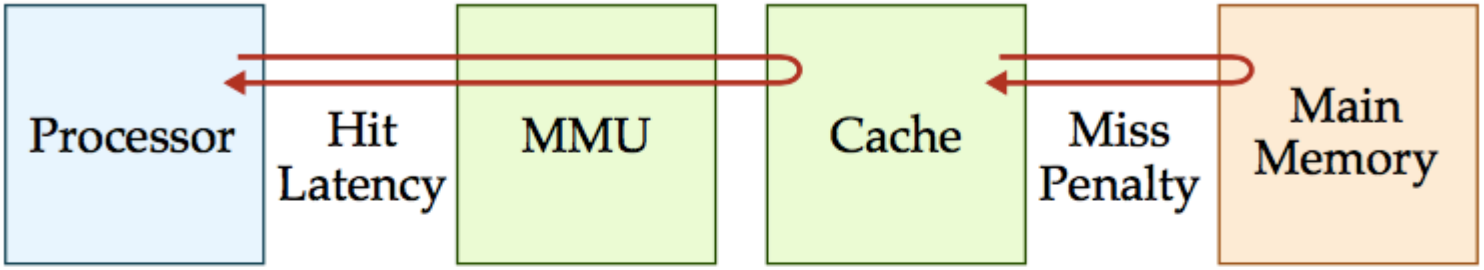
- mem access/sequence depends on program and translation
- time/cycle depends on microarchitecture and implementation
- also known as average memory access latency (AMAL)

Analyze Memory Performance

$$\frac{\text{Avg Cycles}}{\text{Mem Access}} = \frac{\text{Avg Cycles}}{\text{Hit}} + \left(\frac{\text{Num Misses}}{\text{Num Accesses}} \times \frac{\text{Avg Extra Cycles}}{\text{Miss}} \right)$$

- average cycles/hit is called the hit latency; it depends on microarchitecture
- number of misses/number of access is called the miss rate; it depends on microarchitecture
- average extra cycles/miss is called the miss penalty; it depends on microarchitecture, rest of memory system

Analyze Memory Performance

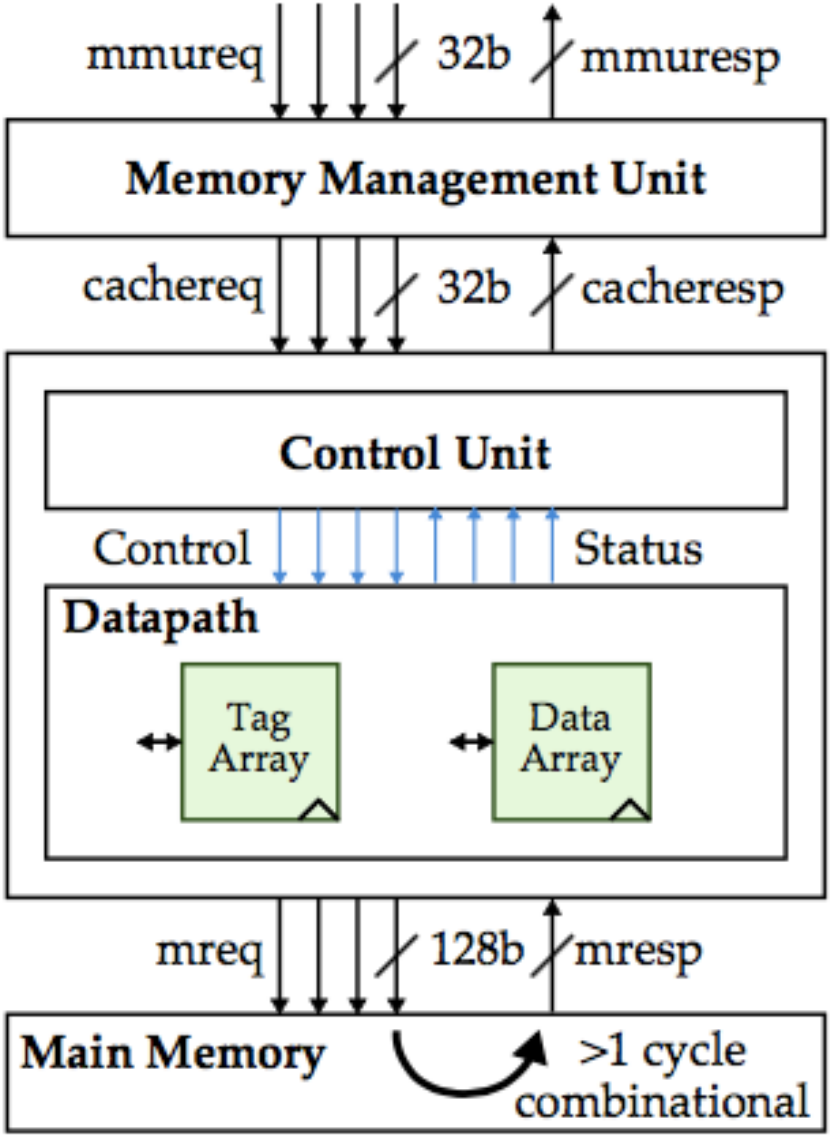


Microarchitecture	Hit Latency	Extra Accesses for Translation
FSM Cache	>1	1+
Pipelined Cache	≈1	1+
Pipelined Cache + TLB	≈1	≈0

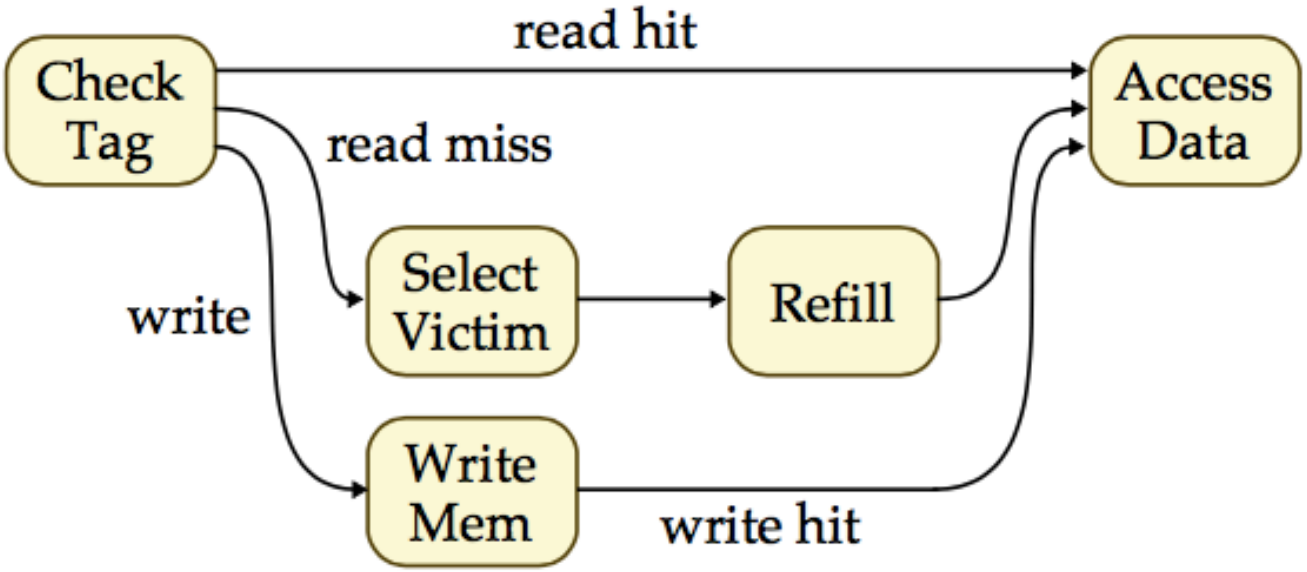
Transactions and Steps, Now for Memory

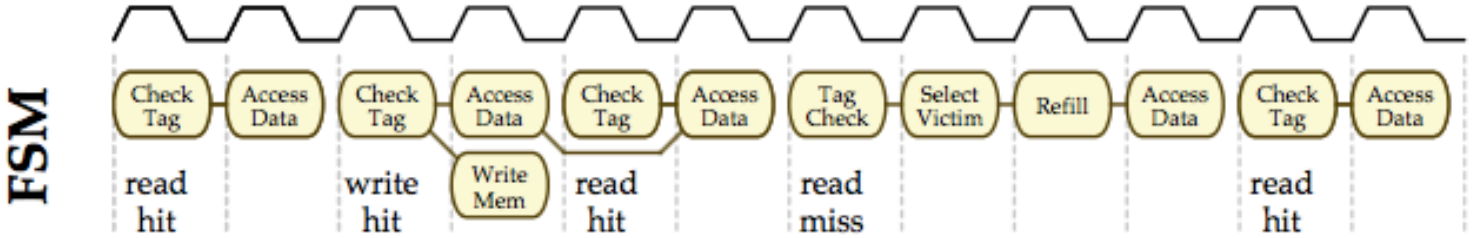
- Executing a memory access involves a sequence of steps
  - check tag: check one or more tags in cache
  - select victim: select victim line from cache using replacement policy
  - evict victim: evict victim line from cache and write victim to memory
  - refill: refill requested line by reading line from memory
  - write mem: write requested word to memory
  - access data: read or write requested word in cache

Memory Microarchitecture Overview

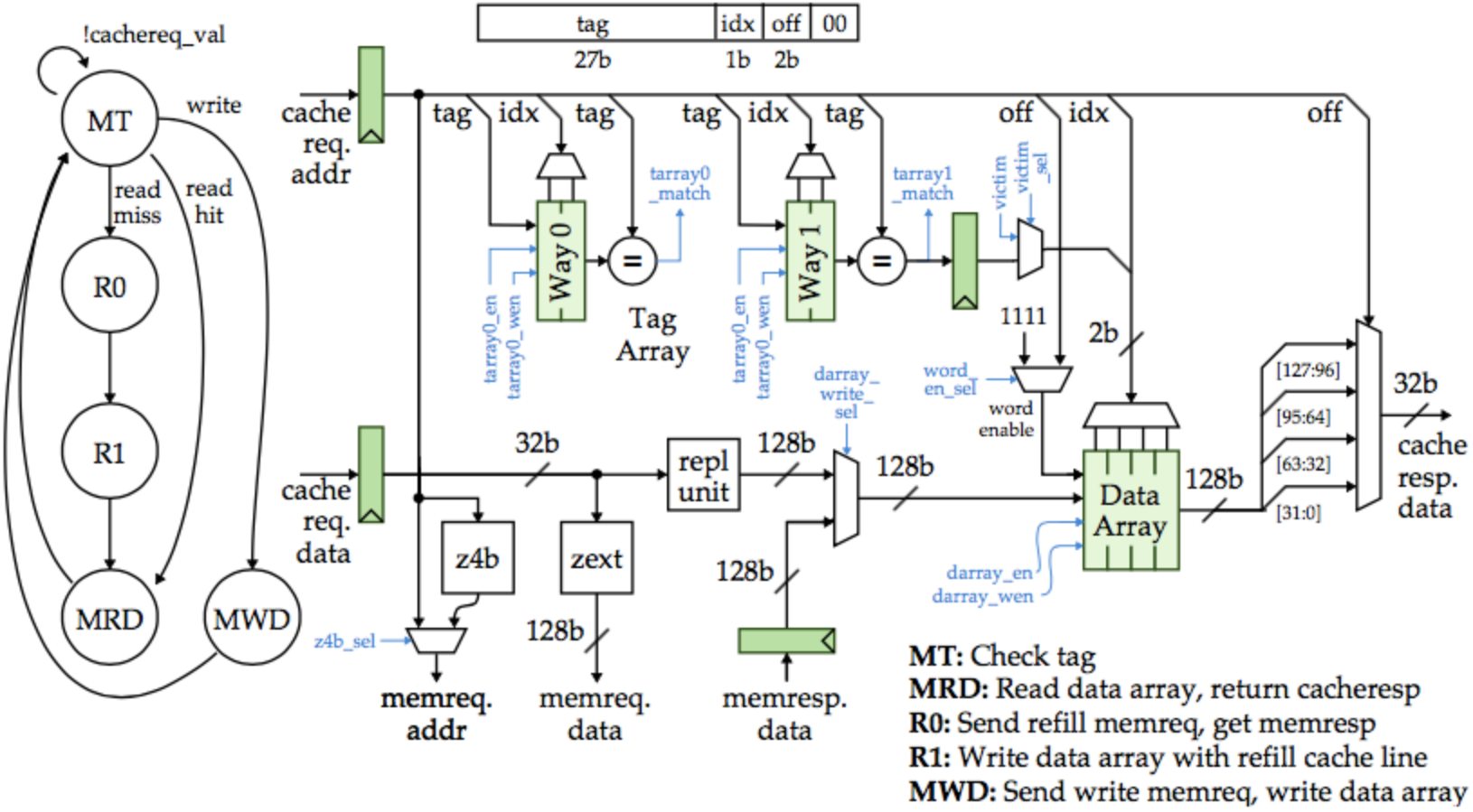


High-level Idea for FSM Cache

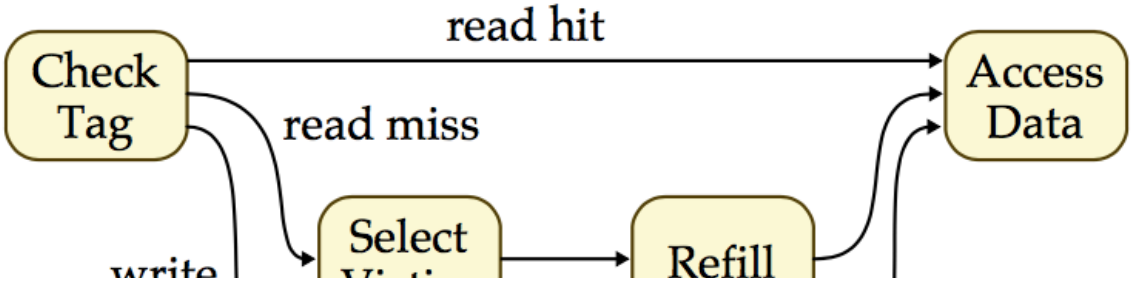


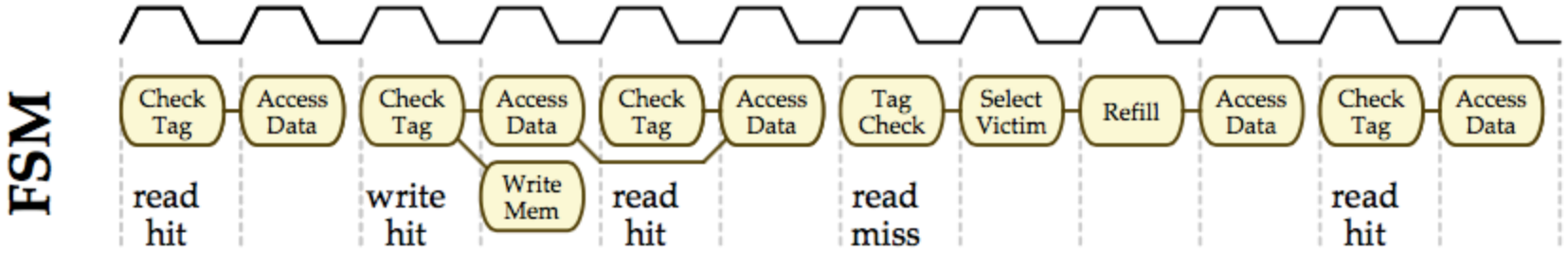


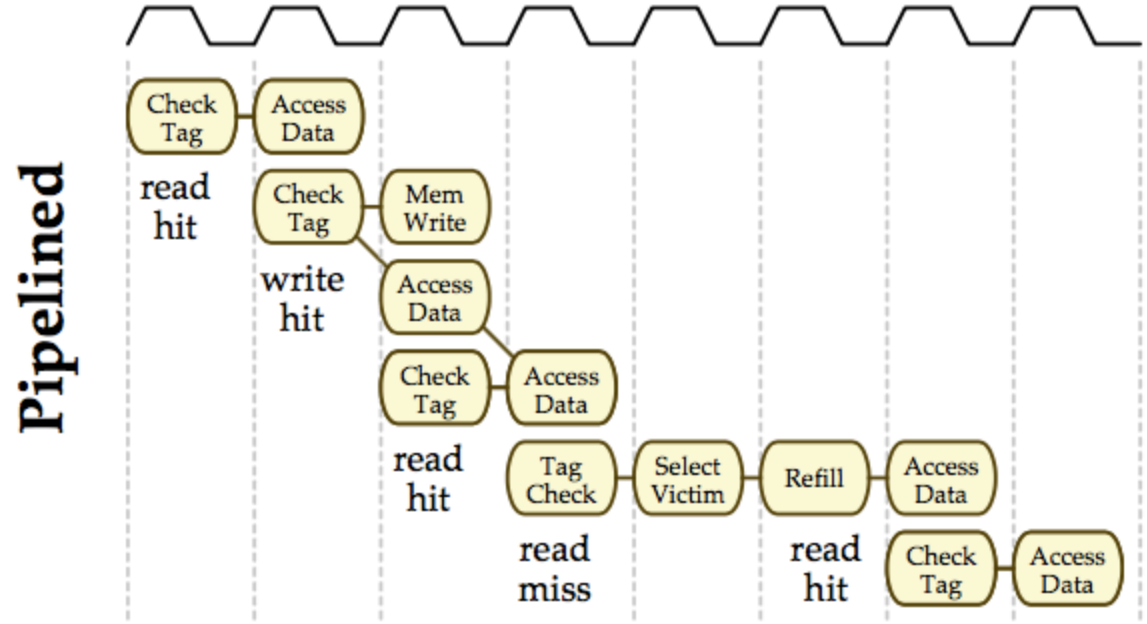
FSM Cache Datapath



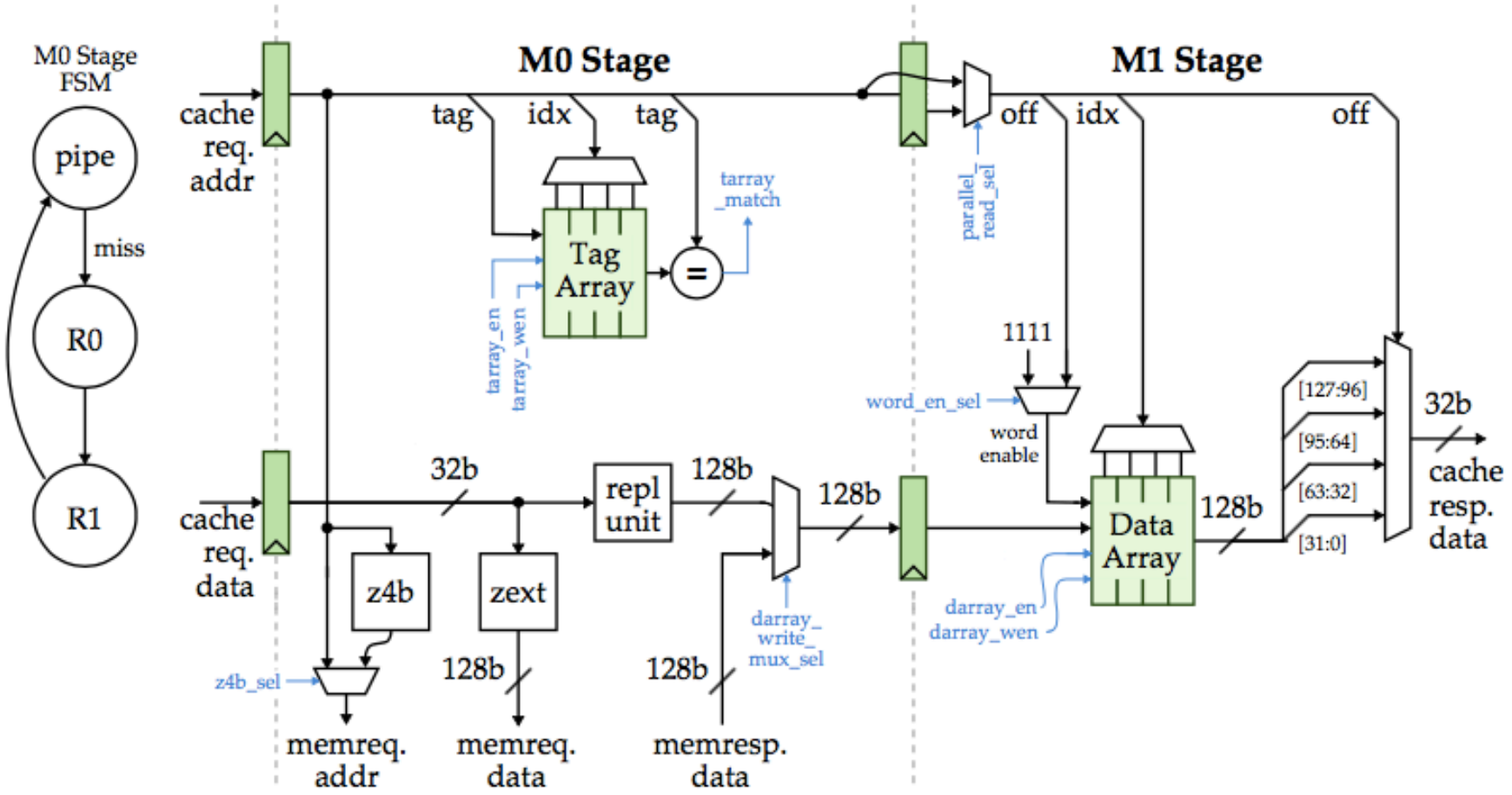
High-level Idea for Pipelined Cache







Pipeline Cache Datapath



Questions?

Comments?

Discussion?

Acknowledgement

Cornell University, ECE 4750