Memory Hierarchy

Chixiao Chen

Breaking News

- The 3 recipients of Turing Awards 2019 is:
  Geoffrey Hinton, Yann Lecun, Yoshua Bengio for their developing conceptual foundations for deep neural networks.

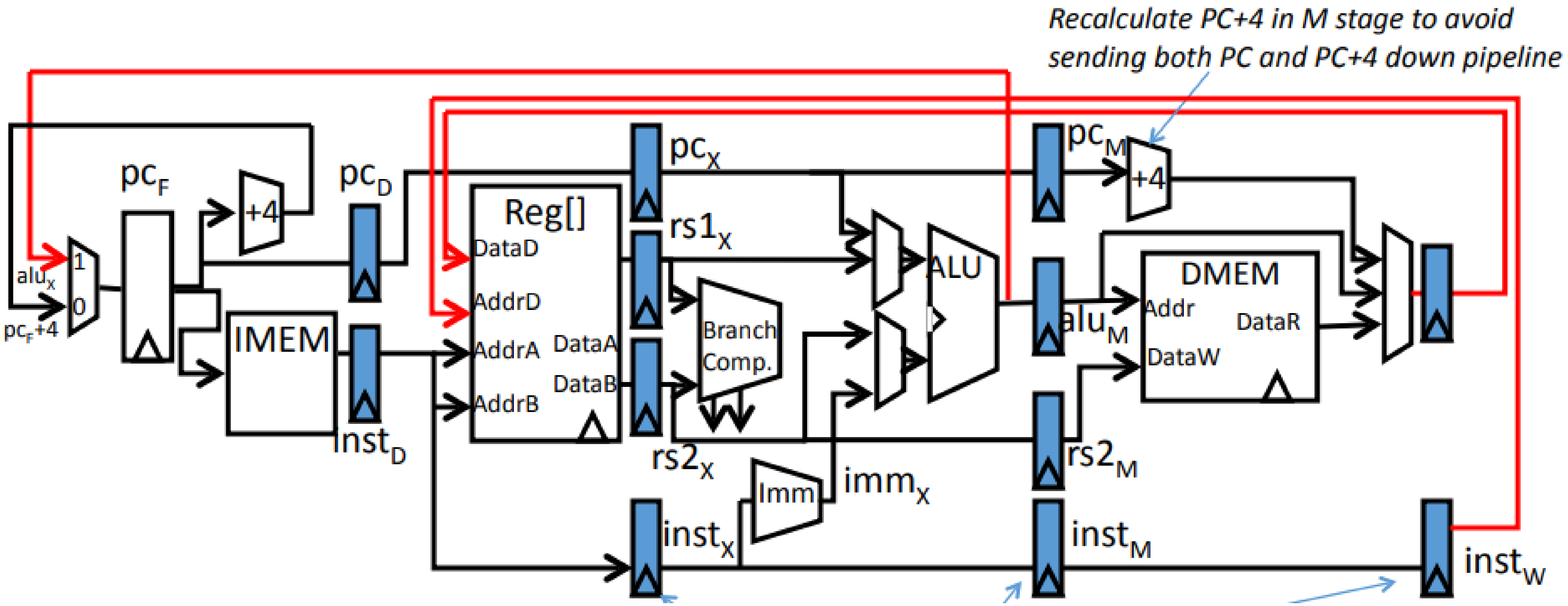
Overview

- Memory Basics
- Memory Hierarchy
  - Cache read
  - Cache Write
- Scratchpad

Memory Basics

We have no pay too much attention on ?

- 32 bit Processor → How big of memory shall we have?

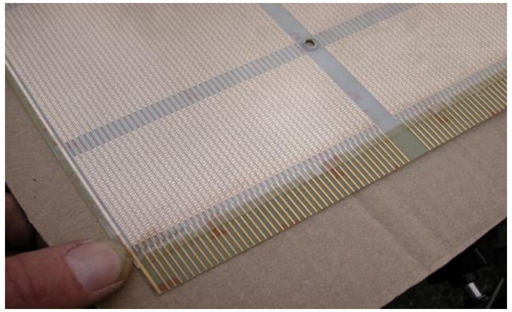


Memory History

- Early Read-Only Memory
  - Punched Cards/ Tapes
  - Capacitors
- Semiconductor Memory
  - Semiconductor memory began to be competitive in early 1970s
    - Intel formed to exploit market for semiconductor memory
    - Early semiconductor memory was Static RAM (SRAM).
  - First commercial Dynamic RAM (DRAM) was Intel 1103
    - 1Kbit of storage on single chip
    - charge on a capacitor used to hold value

Punched cards, From early 1700s through Jaquard Loom, Babbage, and then IBM

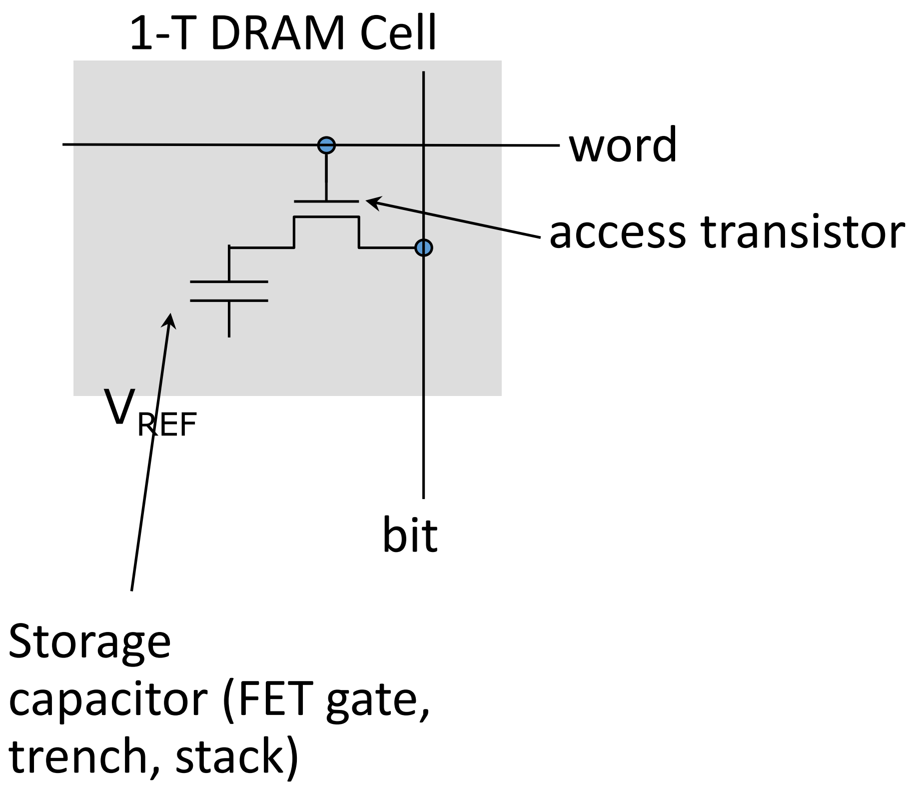
Punched paper tape, instruction stream in Harvard Mk 1

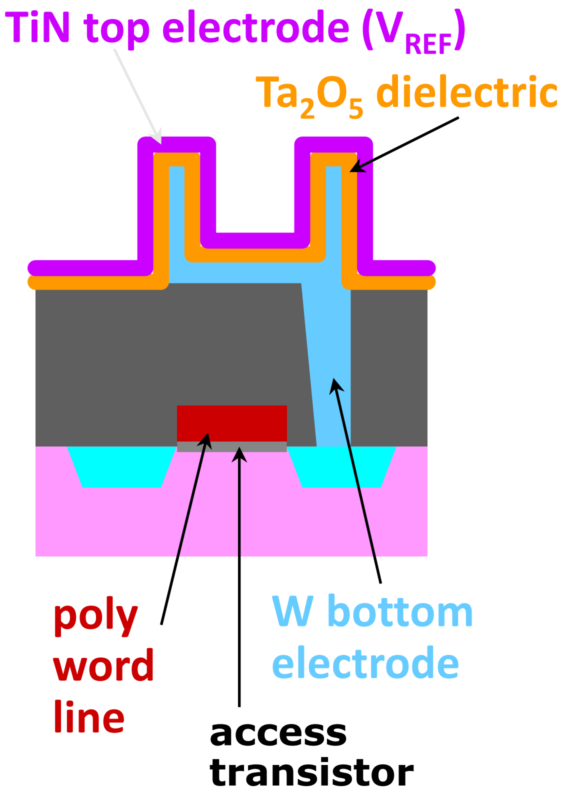


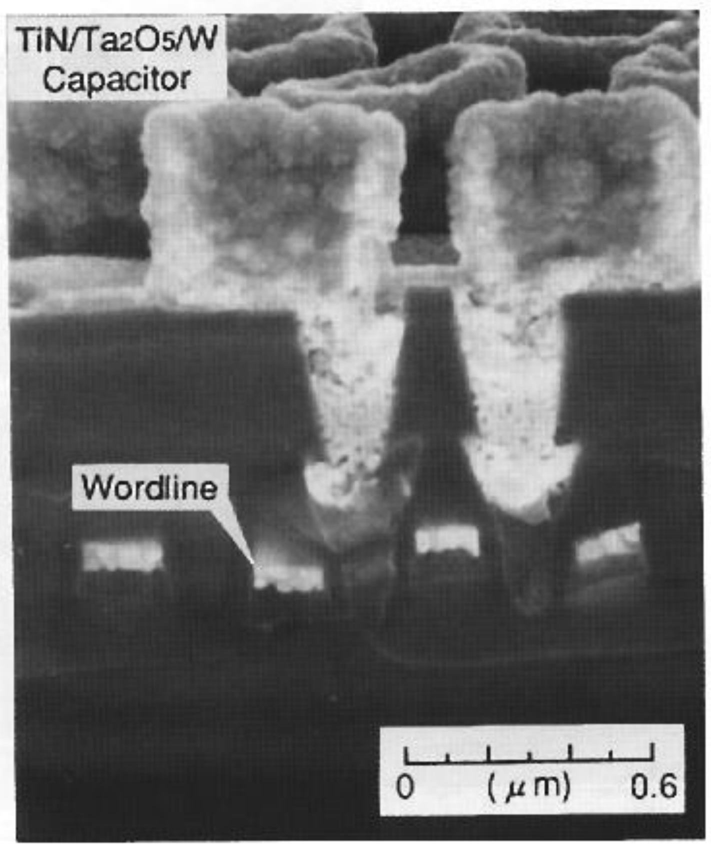
IBM Balanced Capacitor ROS

One-Transistor Dynamic RAM [Dennard, IBM]

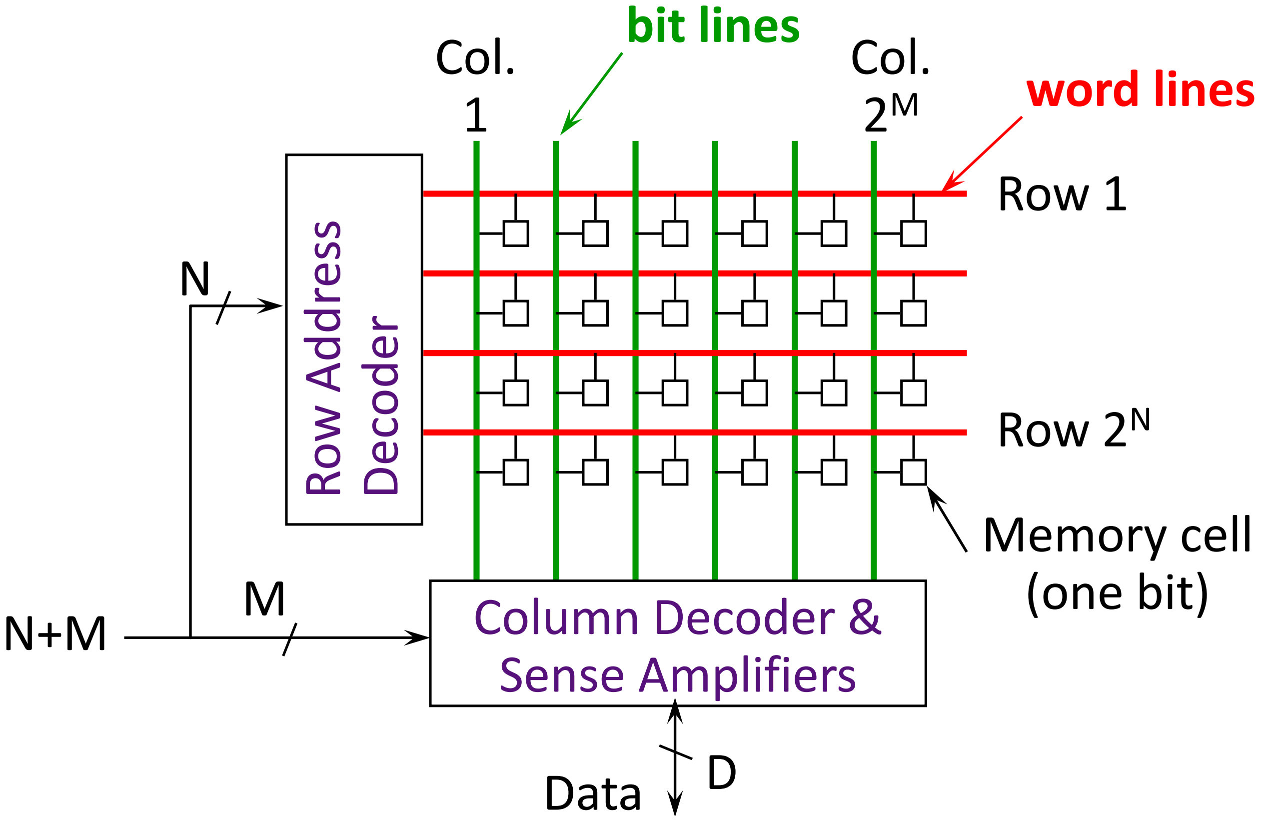
Using one capacitor to store zero / one, using a switch to choose

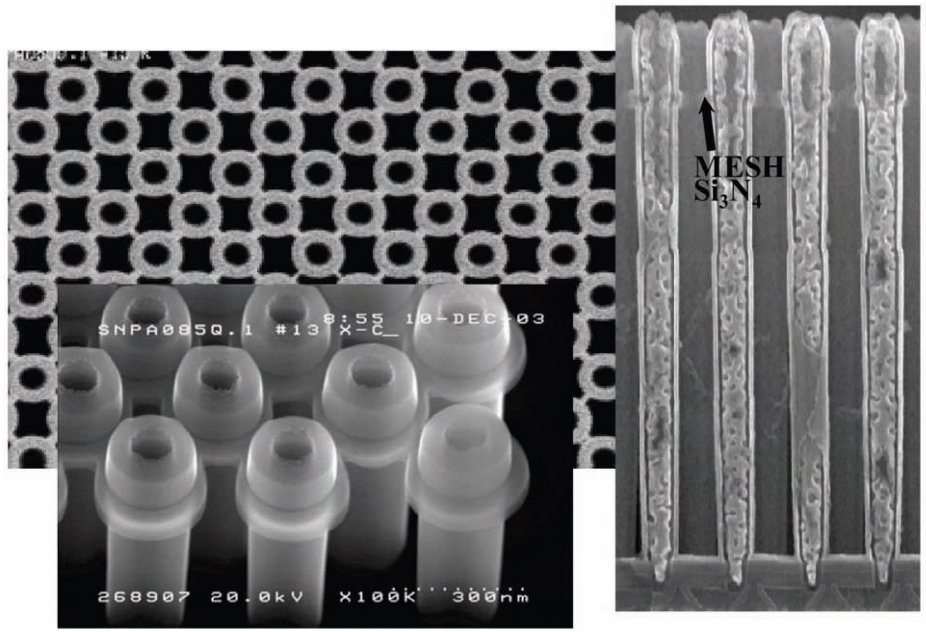






Modern DRAM Architecture / Structure



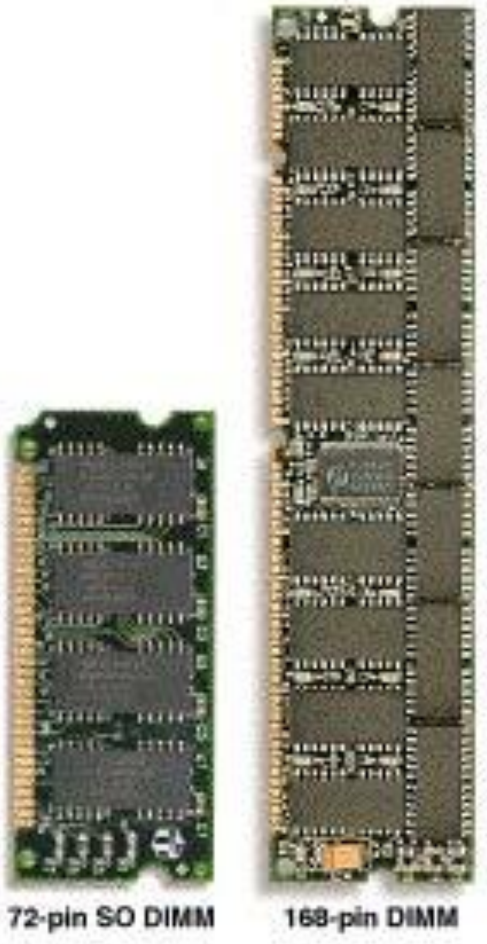


- Problem : DRAM vs. CMOS ?

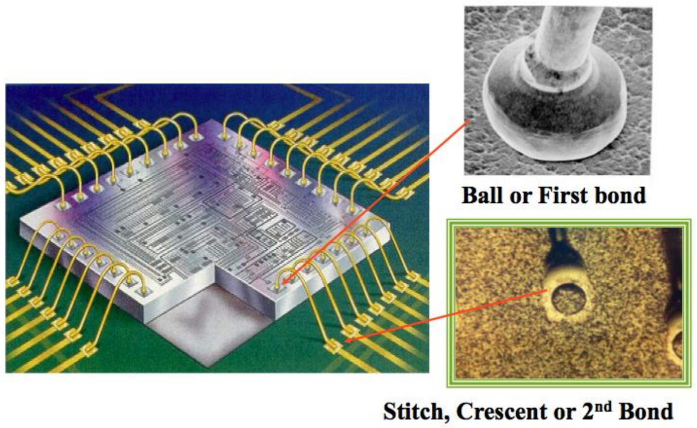
- Bits stored in 2-dimensional arrays on chip

DRAM Packaging I – 2D

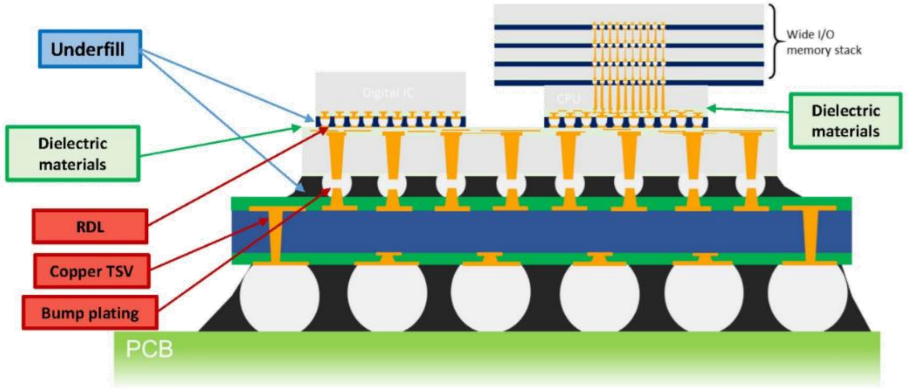
- DIMM (Dual Inline Memory Module) contains multiple chips with clock/control/address signals connected in parallel (sometimes need buffers to drive signals to all chips)
- Data pins work together to return wide word (e.g., 64-bit data bus using 16x4-bit parts)



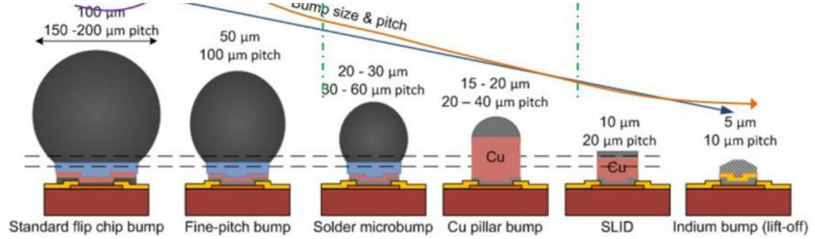
- Single Chip Wire-bonding does not follows Moore’s Law.



DRAM Package II – 3D/2.5D

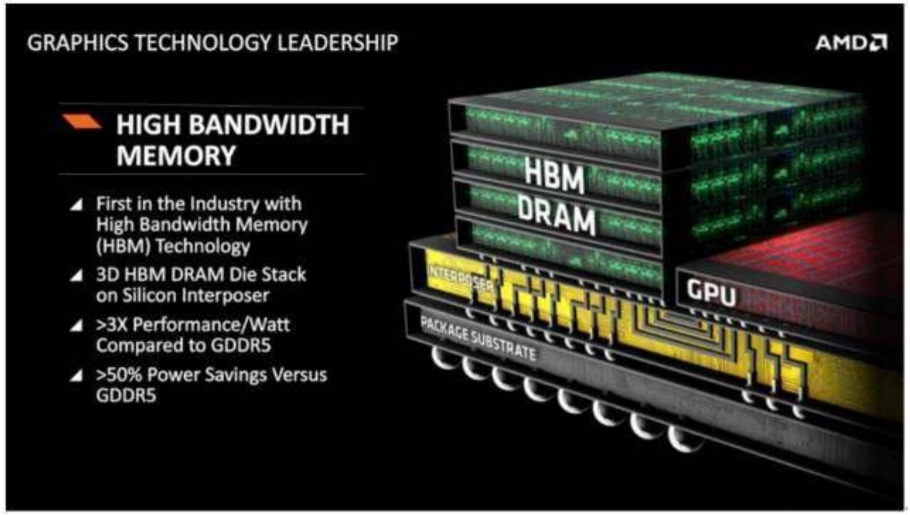


- Flip Chip bonding pad scales as Moore's Law



- TSV (Through silicon vias) techniques further increases the density of transistors, in terms of per area.

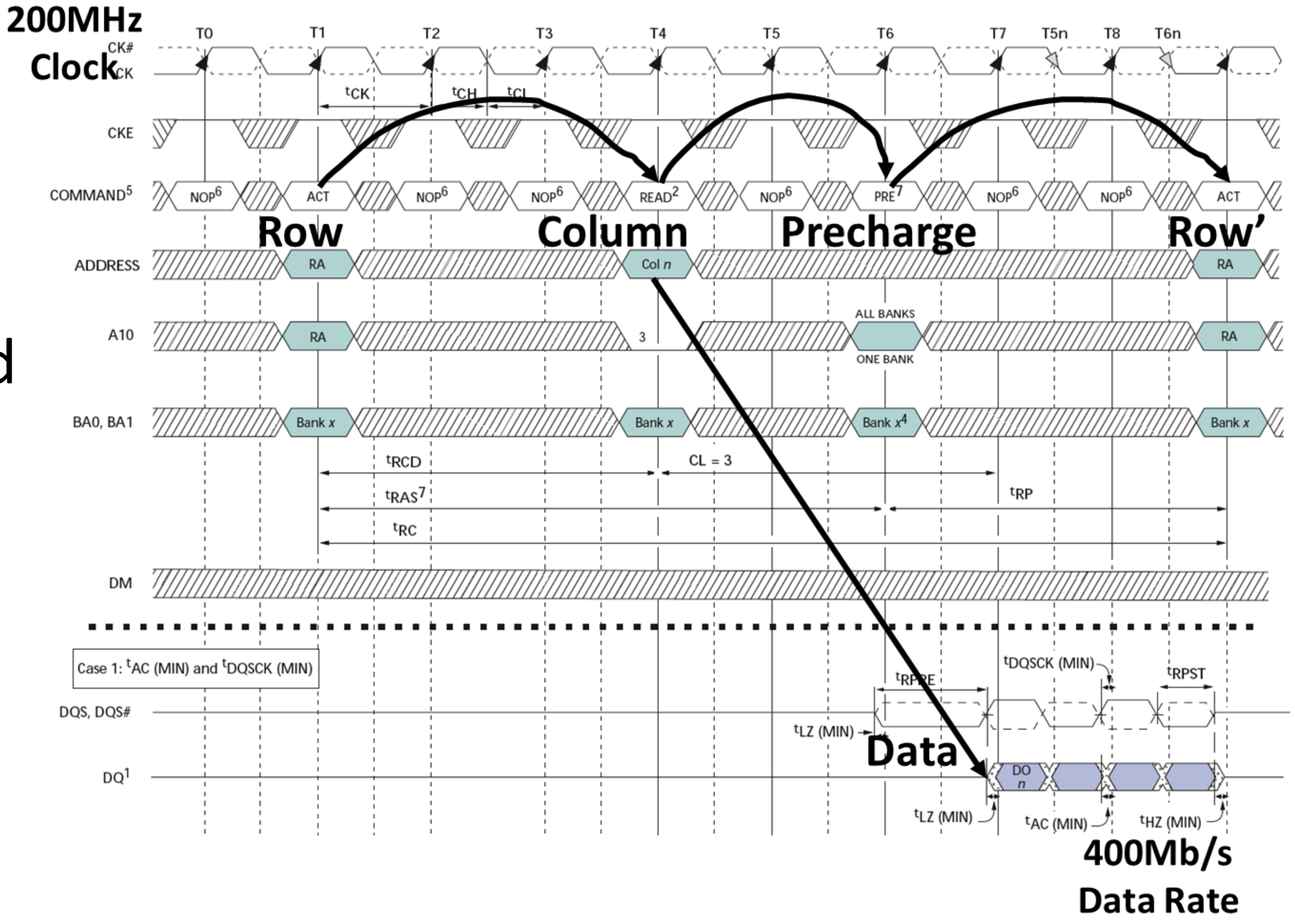
GDDR5	Per Package	HBM
32-bit	Bus Width	1024-bit
Up to 1750MHz (7GBps)	Clock Speed	Up to 500MHz (1GBps)
Up to 28GB/s per chip	Bandwidth	>100GB/s per stack
1.5V	Voltage	1.3V



DDR and Processor-Memory Bottleneck

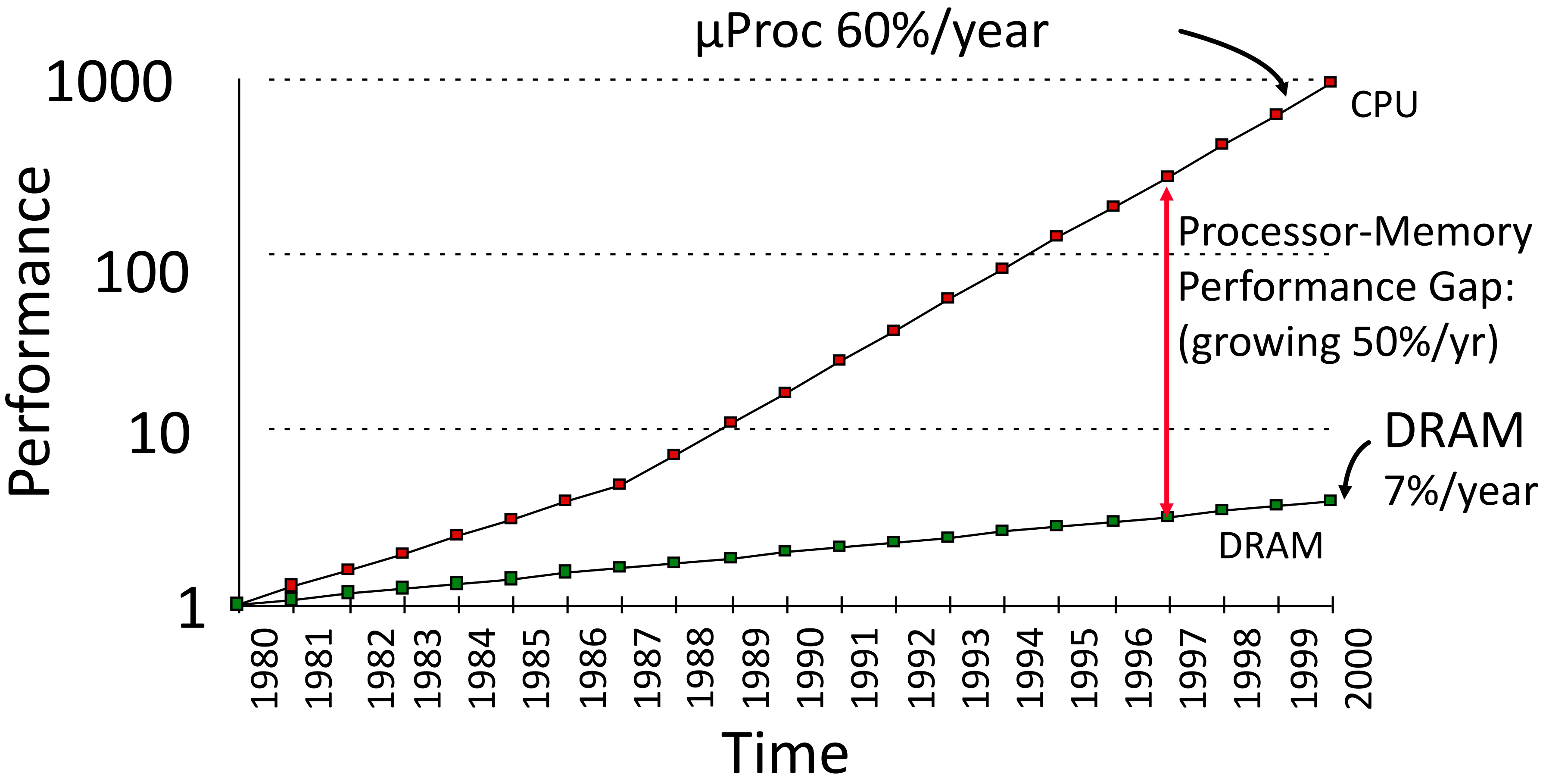
- Double Data Rate

- Performance of high-speed computers is usually limited by memory bandwidth & latency
  - Latency (time for a single access)
  - Bandwidth (number of accesses per unit time)



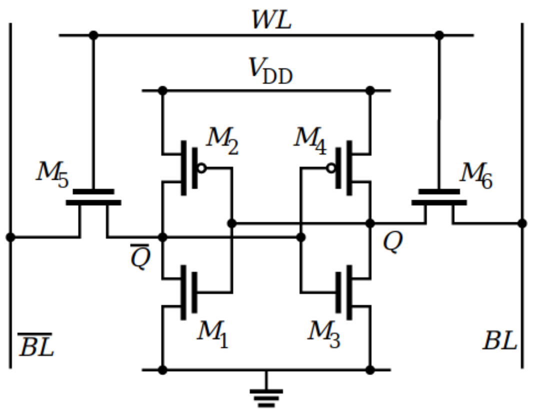
Processor-Memory Gap (Latency)

- 3GHz 4-issue superscalar processor
- 100ns DRAM
- 1200 instruction

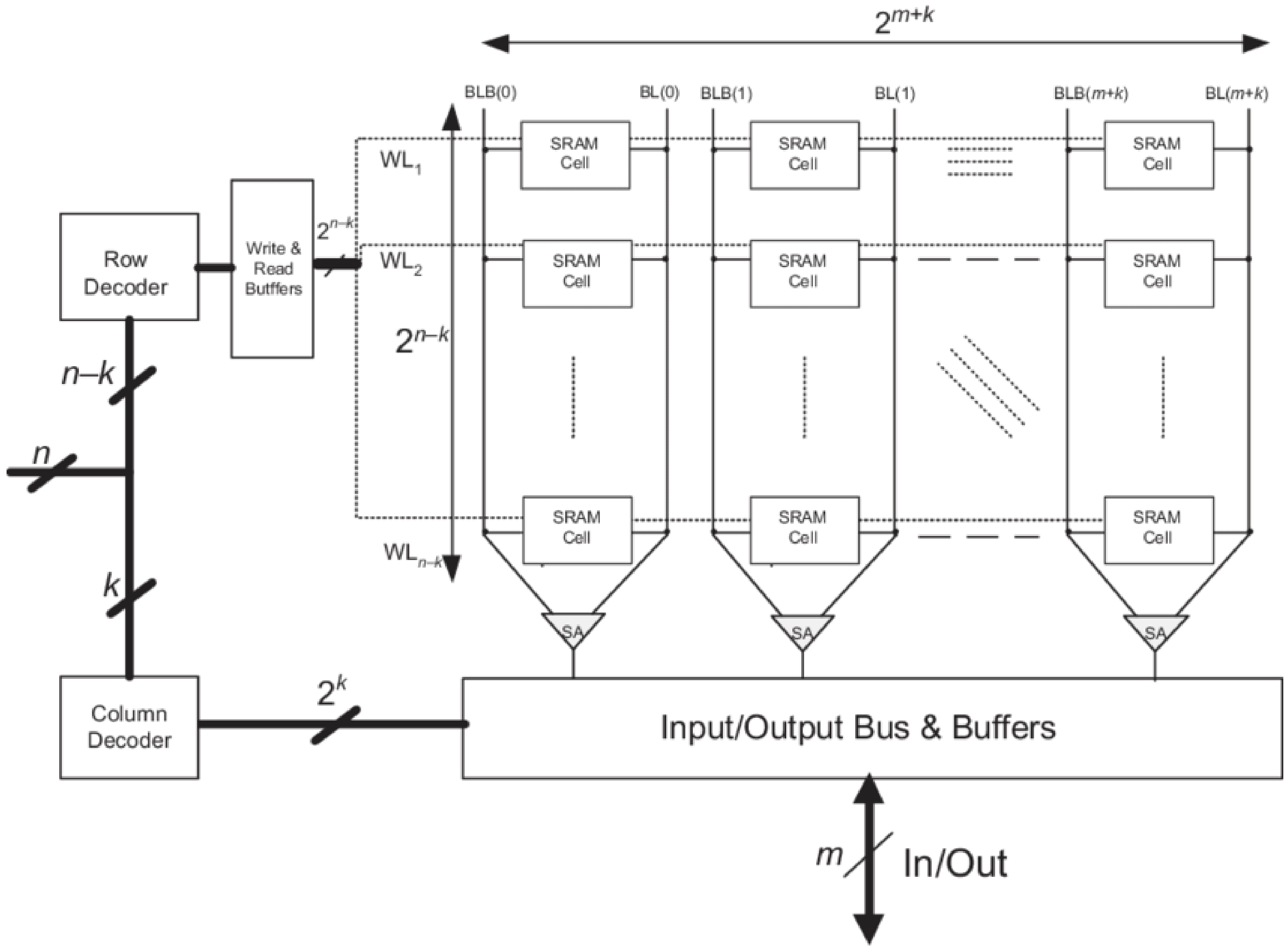


On-Chip Memory

- Static RAM
  - No need to refresh
  - large noise margin, normally in low voltage
  - Disadvantage: Larger area

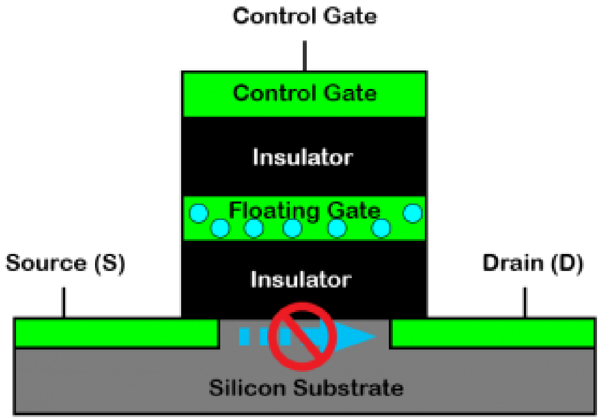


SRAM Memory Cell

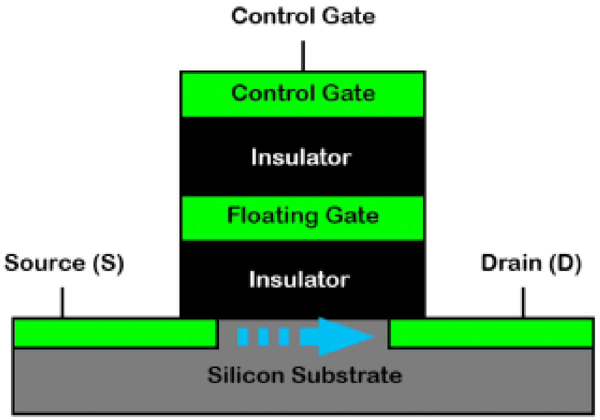


Flash SSD

- Charge in the floating gate

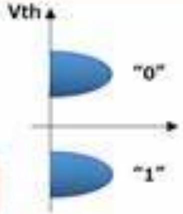
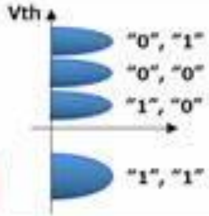
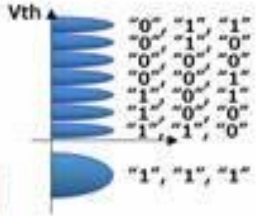


No Current - Floating Gate Programmed



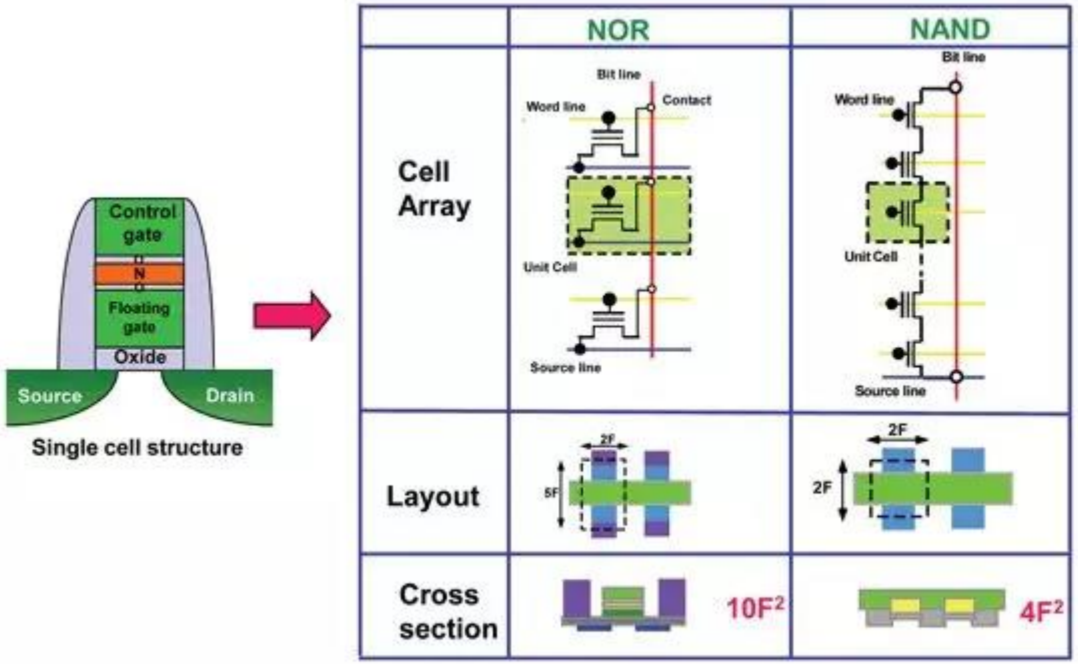
Current Flows - Floating Gate Erased

- Single level Cell vs. Multi-Level Cell vs. Triple-level Cell

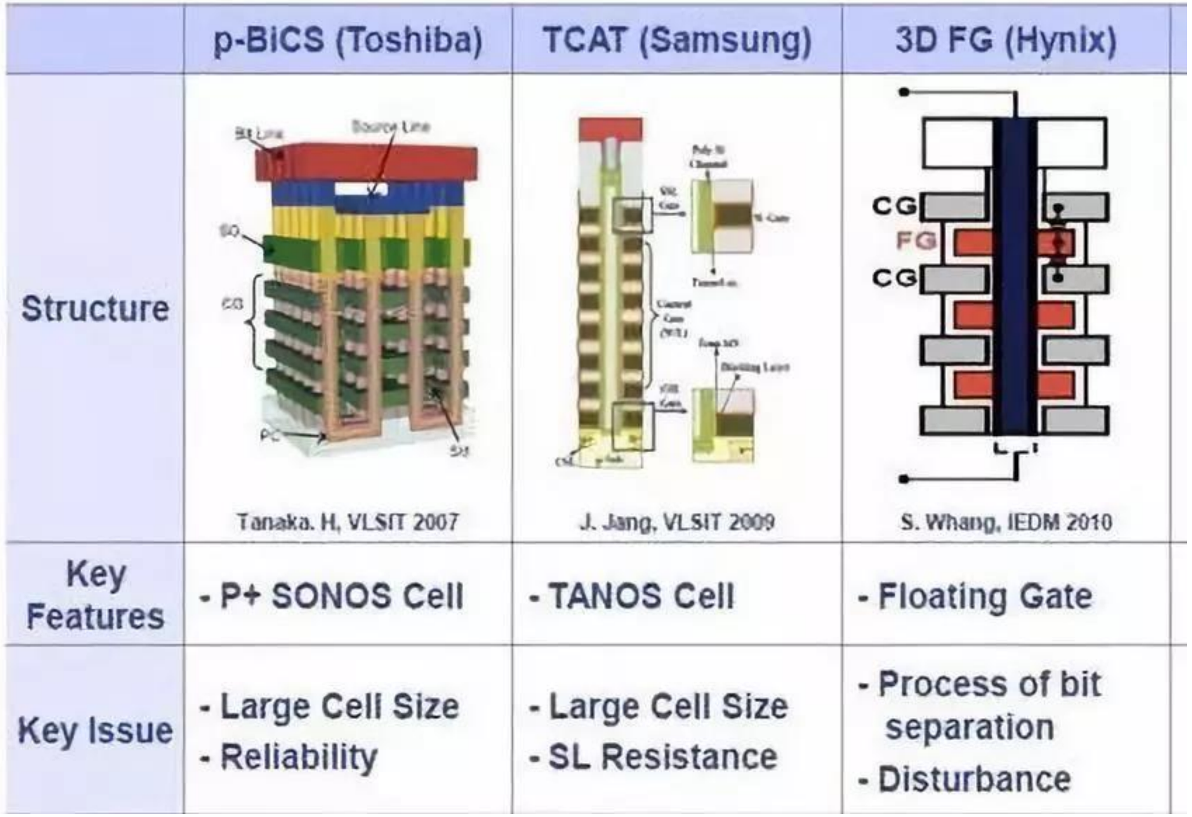
Device Type	Stored Information / Memory Cell	State Count	Vth Distribution of the memory cell
SLC (Single Level Cell)	1 bit / cell	2	
1 bit per cell → Reliable, Higher cost			
MLC (Multi Level Cell)	2 bits / cell	4	
2 bits share same cell → doubled Capacity , less reliable			
TLC (Triple Level Cell)	3 bits / cell	8	
3 bits share same cell → Higher Capacity , poor reliable			

Flash SSD

- NAND Vs. NOR



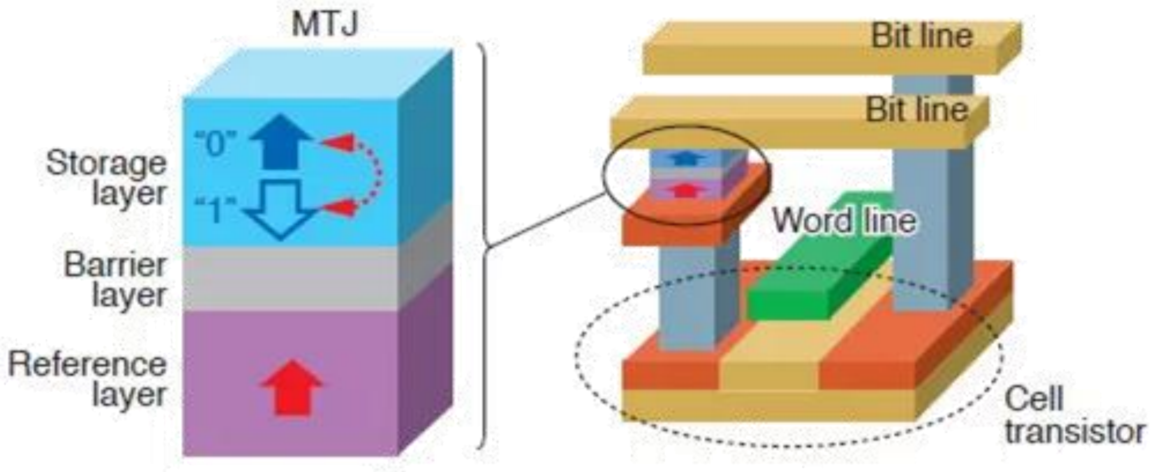
- 3D Flash



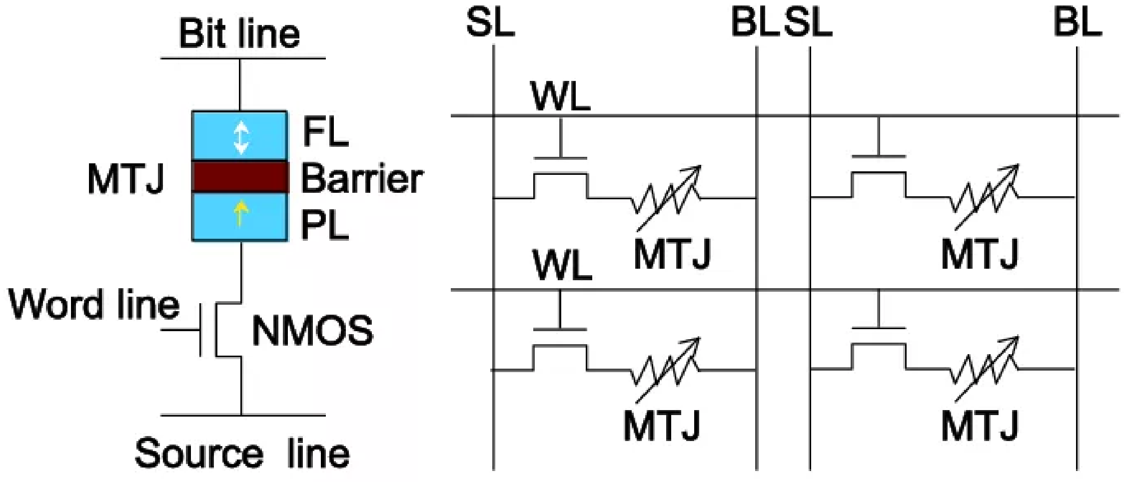
	p-BiCS (Toshiba)	TCAT (Samsung)	3D FG (Hynix)
Structure	Tanaka. H, VLSIT 2007	J. Jang, VLSIT 2009	S. Whang, IEDM 2010
Key Features	- P+ SONOS Cell	- TANOS Cell	- Floating Gate
Key Issue	- Large Cell Size - Reliability	- Large Cell Size - SL Resistance	- Process of bit separation - Disturbance

Future / More advanced Memory

- Volatile and Non-volatile Memory
- As fast/dense as DRAM, as CMOS compatible as SRAM, as Non-volatile as Flash?



- STT – MRAM
  (Spin Transfer Torque Magnetic RAM)
- Magnetic Tunnel Junction



(a) Cell (b) Array

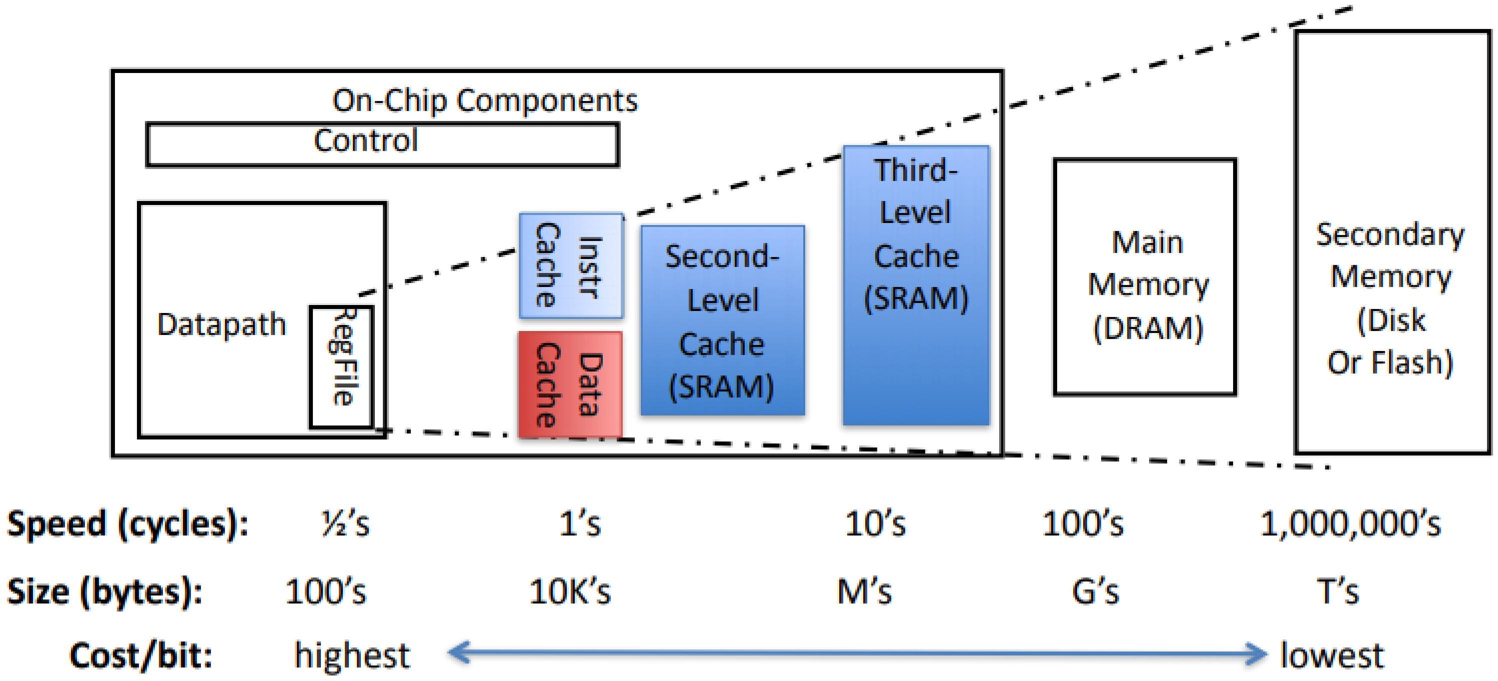
Why we need so many memory?

Hierarchy

Memory Hierarchy

- As much as possible
- As fast as possible
- As cheap as possible

- How ?
- Locality

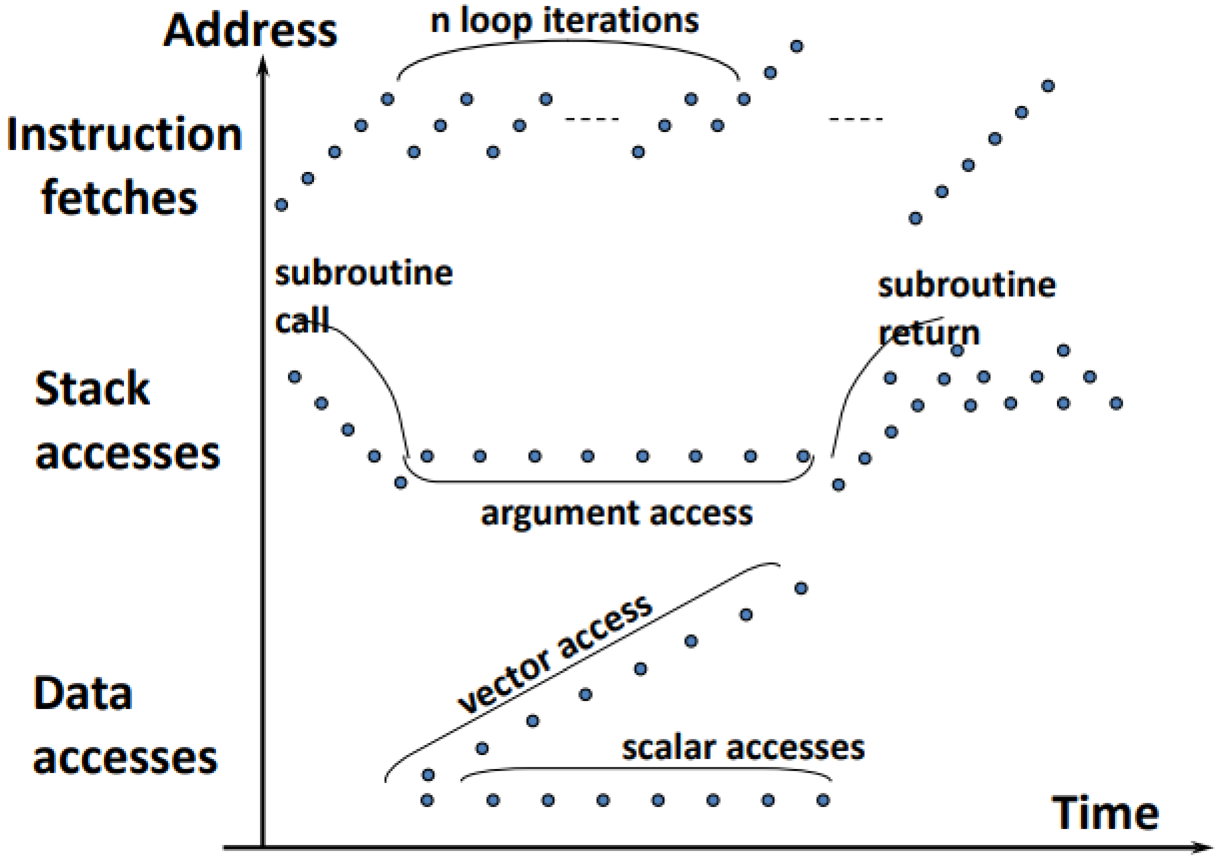


Locality

- Temporal Locality (locality in time)
  - Go back to same book on desktop multiple times
  - If a memory location is referenced, then it will tend to be referenced again soon

- Spatial Locality (locality in space)
  - When go to book shelf, pick up multiple books from same shelf since library stores related books together
  - If a memory location is referenced, the locations with nearby addresses will tend to be referenced soon

Locality



Find examples with temporal / spatial locality in a common program.

Caches ($)

- Exploit temporal locality by remembering the contents of recently accessed locations.
- Exploit spatial locality by fetching blocks of data around recently accessed locations.

- Caches copy some useful data blocks from memory
    - Small but fast

Cache Read

- Look at Processor Address, check whether data is in cache, Then either

Found in cache
a.k.a. HIT

Return copy
of data from
cache

Not in cache
a.k.a. MISS

Read block of data from
Main Memory

Wait …

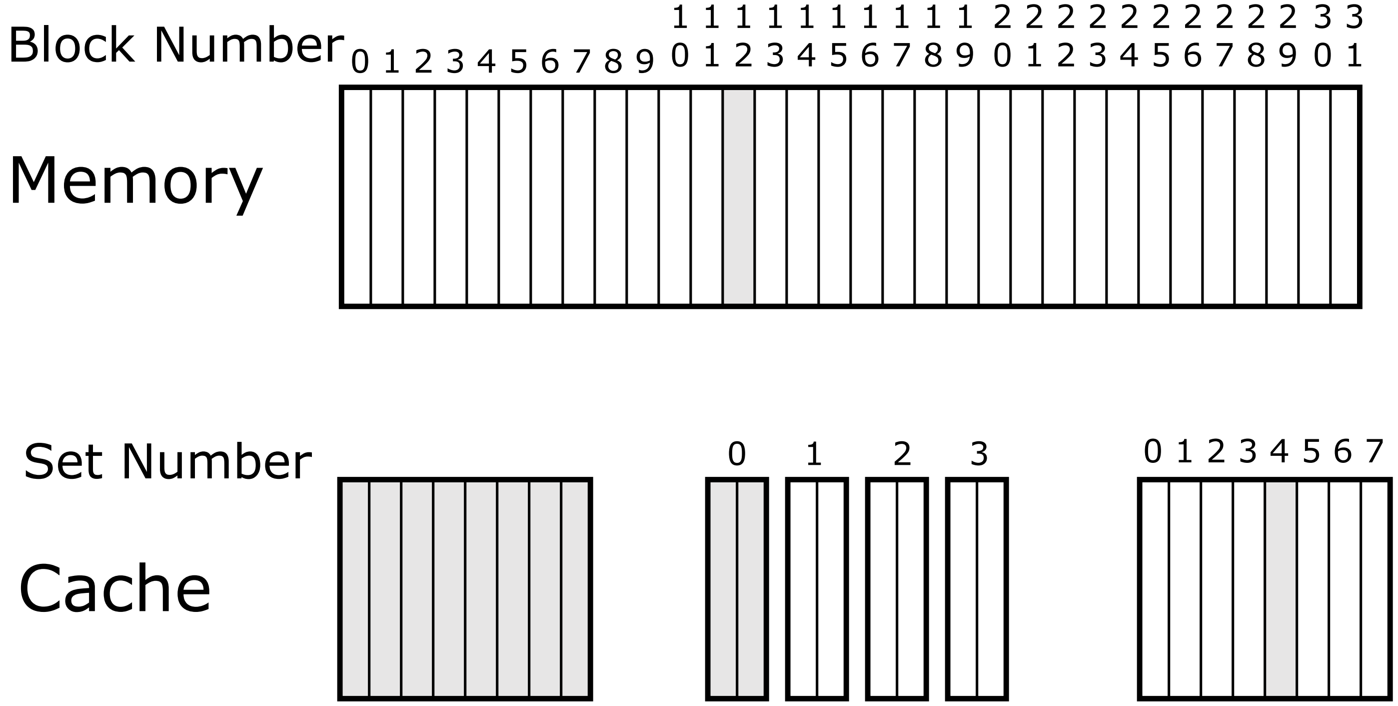
Return data to processor
and update cache

Q: Which line do we replace?

Placement Policy

- Assume Memory is 4x larger than cache

- Where can I put No.12 block in memory to cache?



Fully Associative
anywhere

(2-way) Set Associative
anywhere in set 0
(12 mod 4)

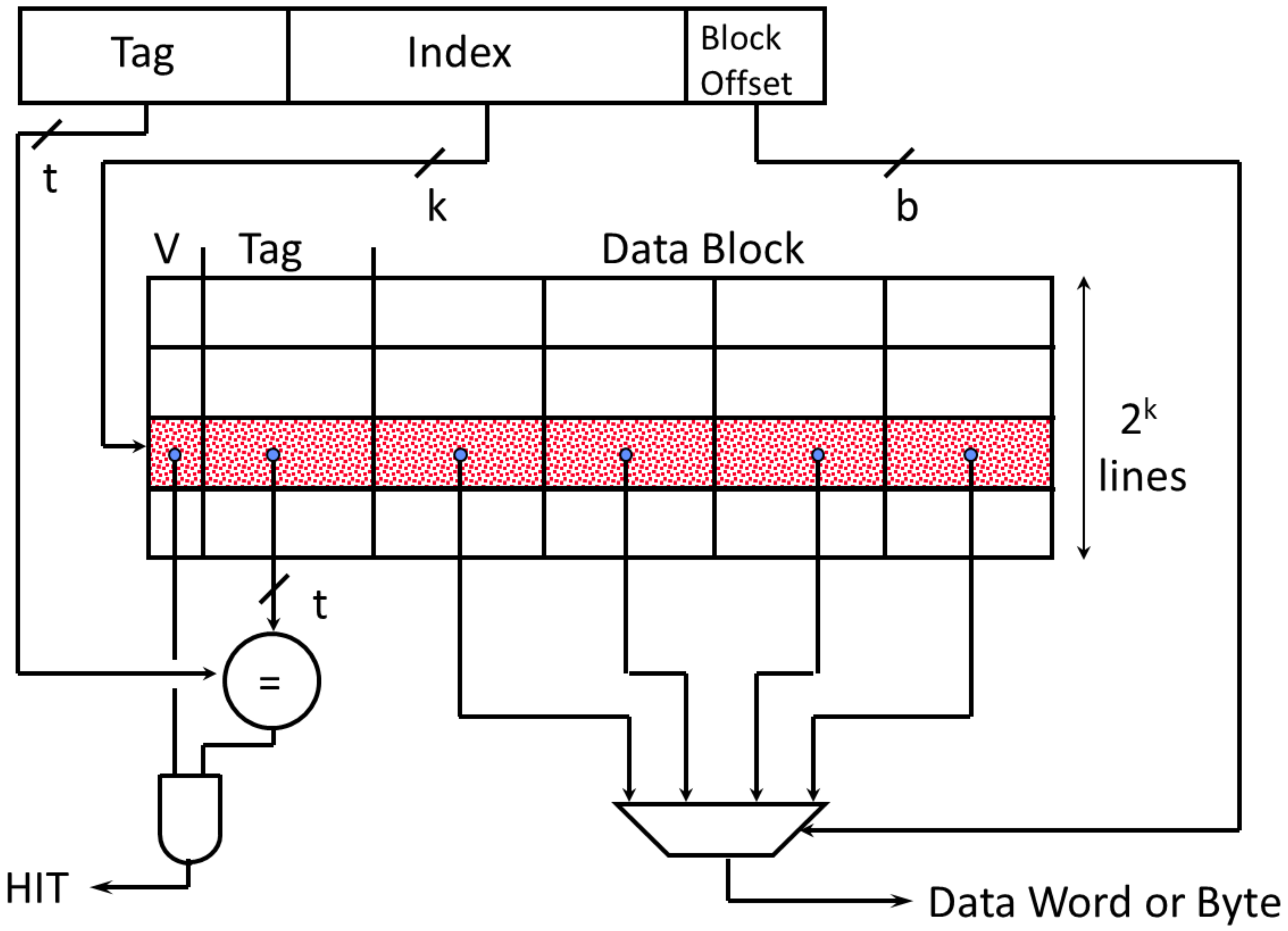
Direct Mapped
only into block 4
(12 mod 8)

block 12
can be placed

Direct Map Cache

- How to check hit?
- Address Fields in Cache controller
  - Index: Selects which set
  - Tag: Remaining portion of processor address
  - Block Offset: Byte address within block

- How to calculate?



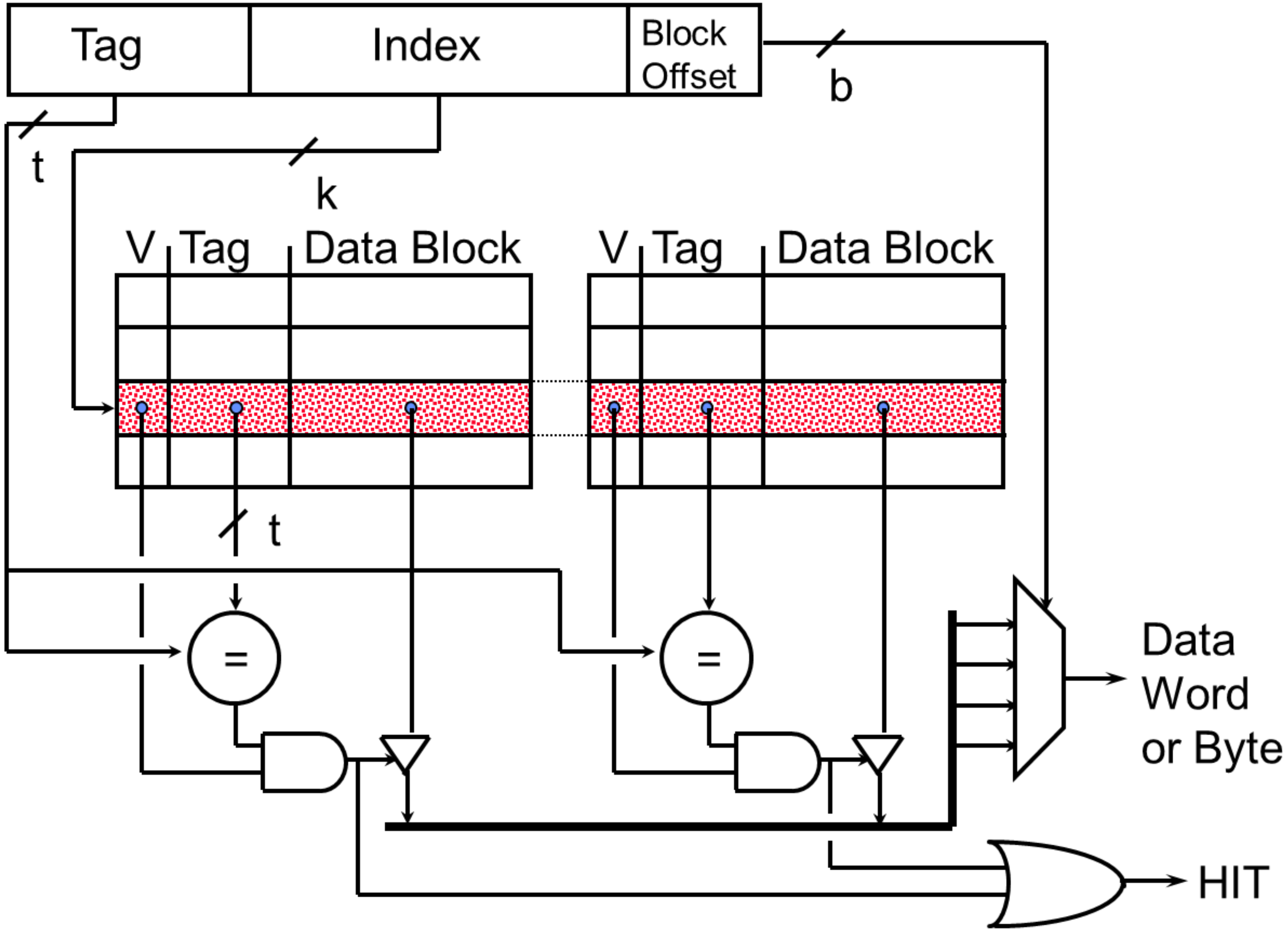
Direct Mapped Cache

Tag	*Index*	*Byte Offset*
Mem Block Within $ Block	Block Within $	Byte Within Block

- In example, block size is 4 bytes (1 word)
- Memory and cache blocks always the same size, unit of transfer between memory and cache
- # Memory blocks >> # Cache blocks
- 16 Memory blocks = 16 words = 64 bytes => 6 bits to address all bytes
- 4 Cache blocks, 4 bytes (1 word) per block
- 4 Memory blocks map to each cache block
- Memory block to cache block, aka index: middle two bits
- Which memory block is in a given cache block, aka tag: top two bits

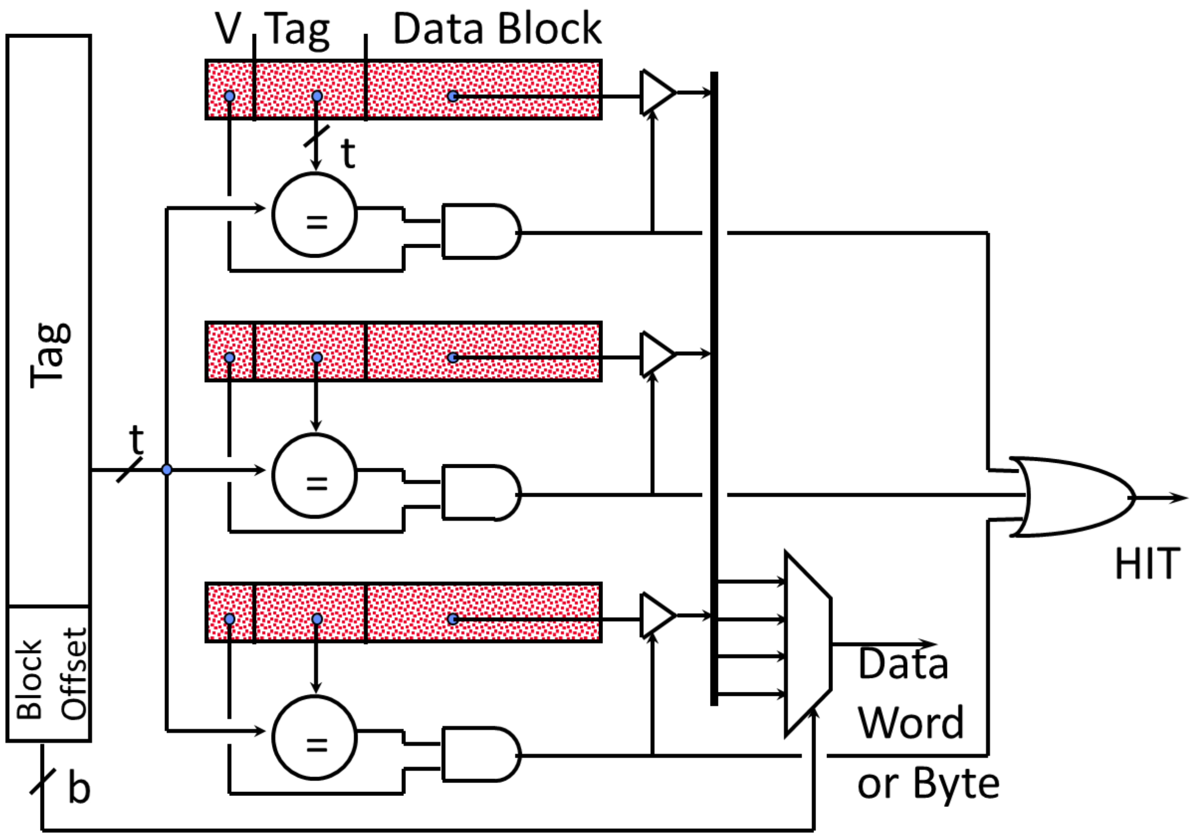
2-Way Set-Associative Cache

- "N-way Set Associative"
- N places for a block
- Number of sets = number of blocks / N
- N comparators
  - Fully Associative: N = number of blocks
  - Direct Mapped: N = 1



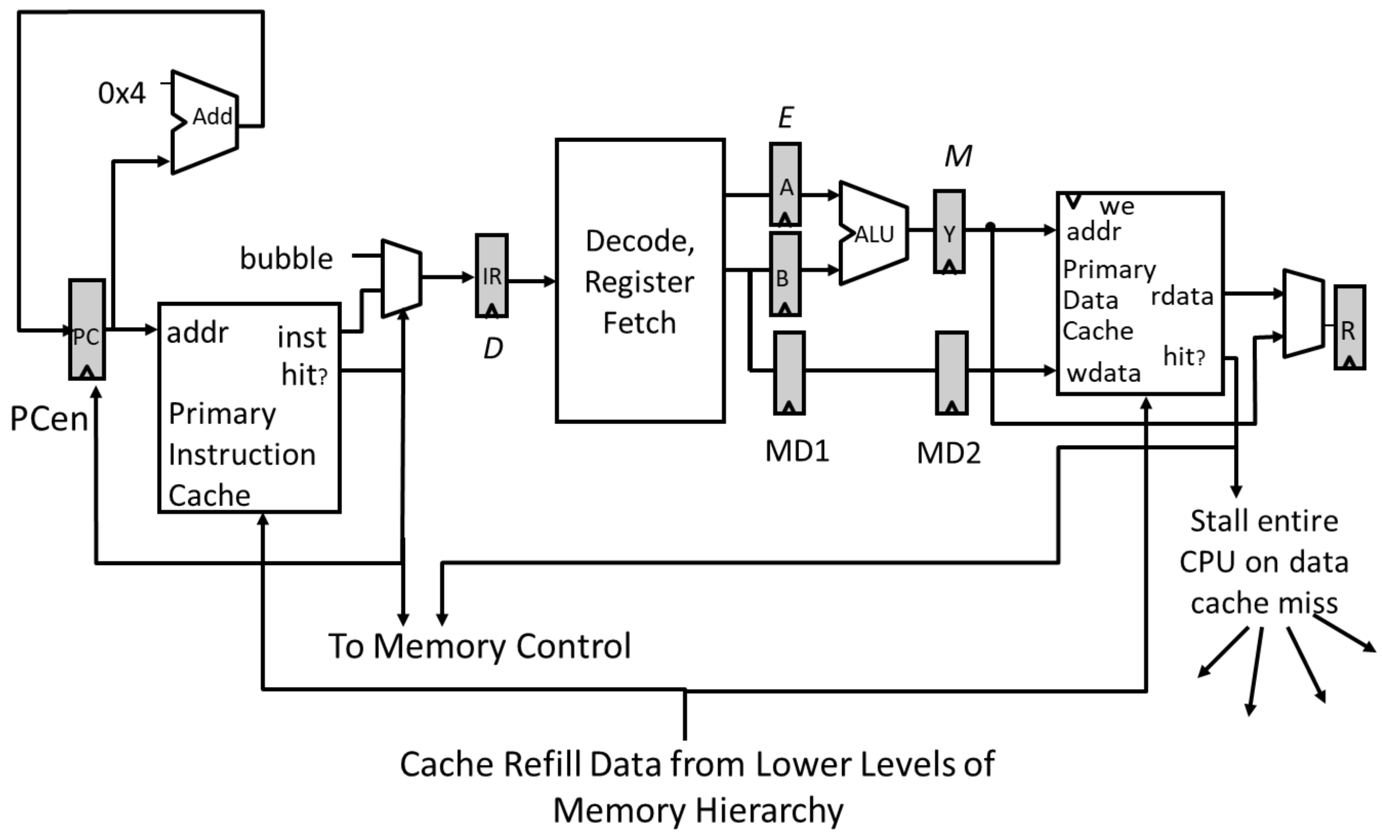
Fully Associative Cache

- The HIT check needs much time
- Hardware is not efficient



Cache – Memory for Pipeline Architecture

- To improve performance:
  - reduce the hit time
  - reduce the miss rate
  - reduce the miss penalty
- *What is best cache design for 5-stage pipeline?*



AMAT = Time for a hit + Miss rate × Miss penalty

Causes of Cache Misses: The 3 C's

Compulsory: first reference to a line (a.k.a. cold start misses)

- *misses that would occur even with infinite cache*

Capacity: cache is too small to hold all data needed by the program

- *misses that would occur even under perfect replacement policy*

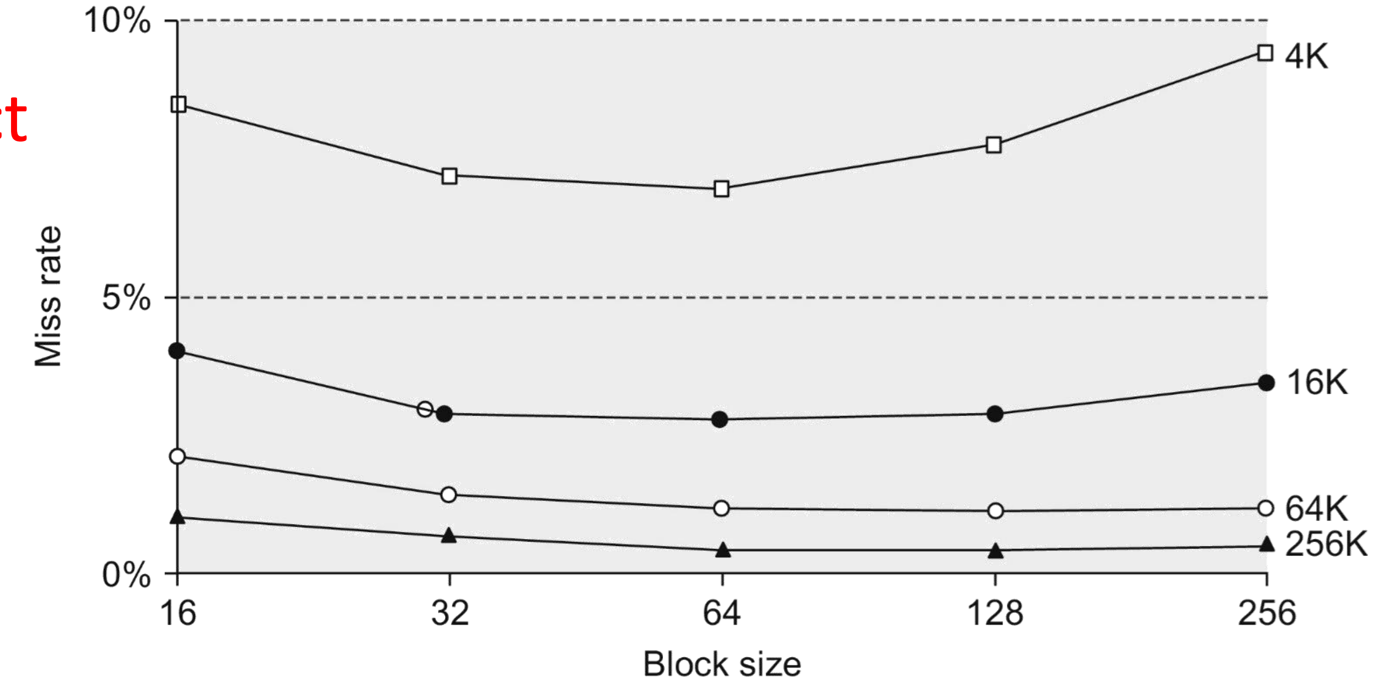
Conflict: misses that occur because of collisions due to line-placement strategy

- *misses that would not occur with ideal full associativity*

Cache Parameter Effects on Performance

- Larger cache size
  + reduces capacity and conflict misses
  - hit time will increase

- Higher associativity
  + reduces conflict misses
  - may increase hit time



- Larger line size
  + reduces compulsory and capacity (reload) misses
  - increases conflict misses and miss penalty

Cache Write

- What's the difference between Cache Read and Write? (if hit?)

Found in cache
a.k.a. HIT

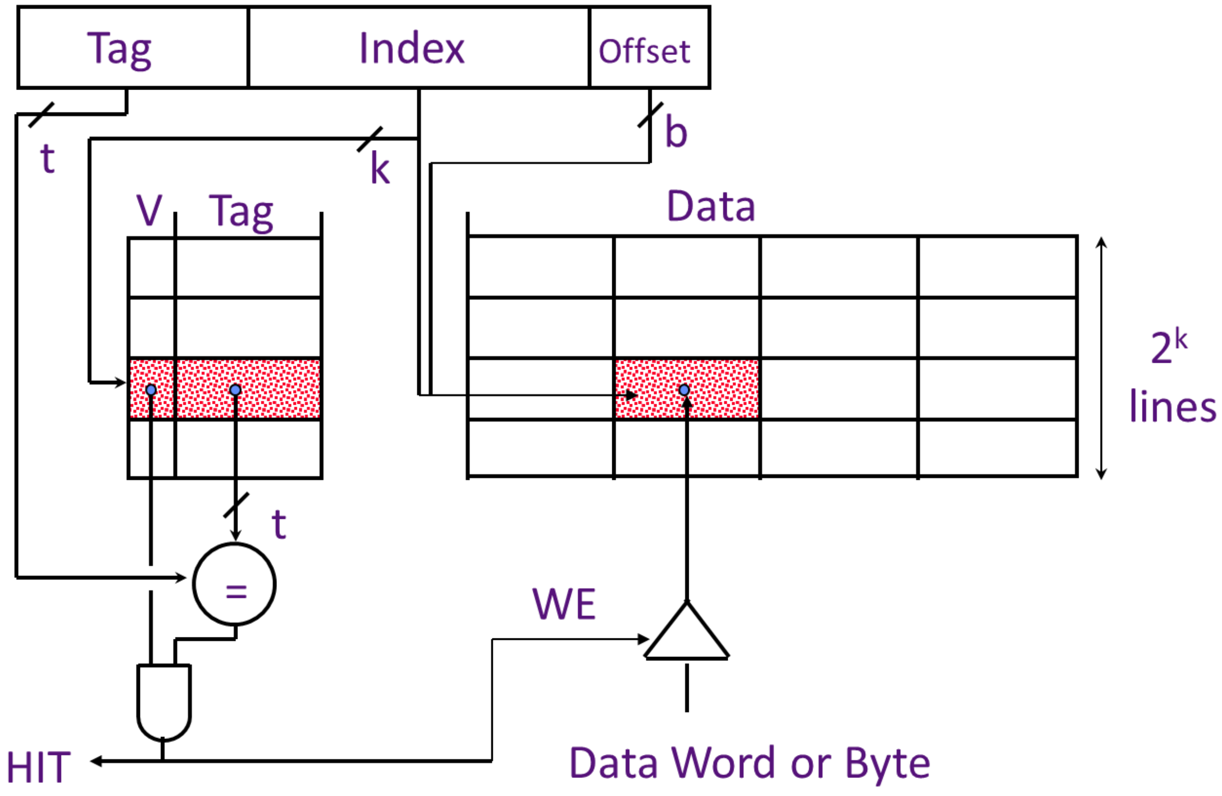
Target: change the value
in memory.

Question: Do we need to change
the value in cache?

- Combination of WAR?

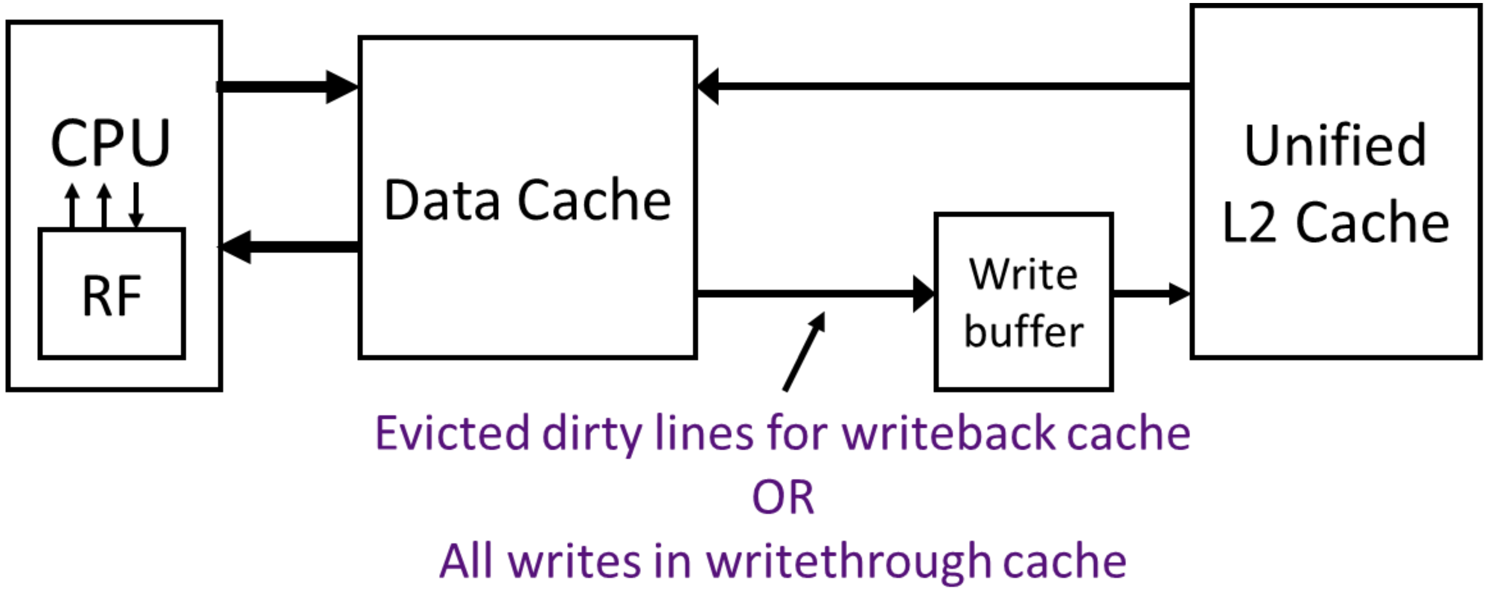
Write Policy If Hit

- write through: write both cache & memory
  - Generally higher traffic but simpler pipeline & cache design
- write back: write cache only, memory is written only when the entry is evicted
  - A dirty bit per line further reduces write-back traffic
  - Must handle 0, 1, or 2 accesses to memory for each load/store



Write Policy If Miss

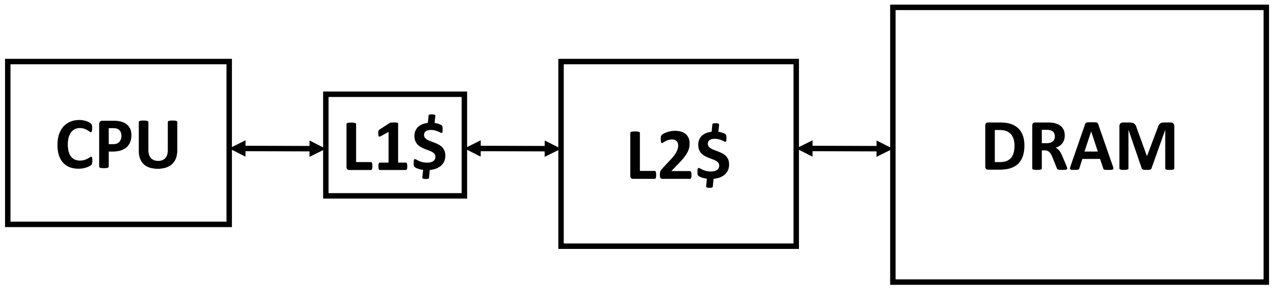
- Cache miss:
  - **no write allocate**: only write to main memory
  - **write allocate** (aka fetch on write): fetch into cache
- Write Buffer to reduce the Miss Penalty



Multilevel Caches

Problem: A memory cannot be large and fast

Solution: Increasing sizes of cache at each level

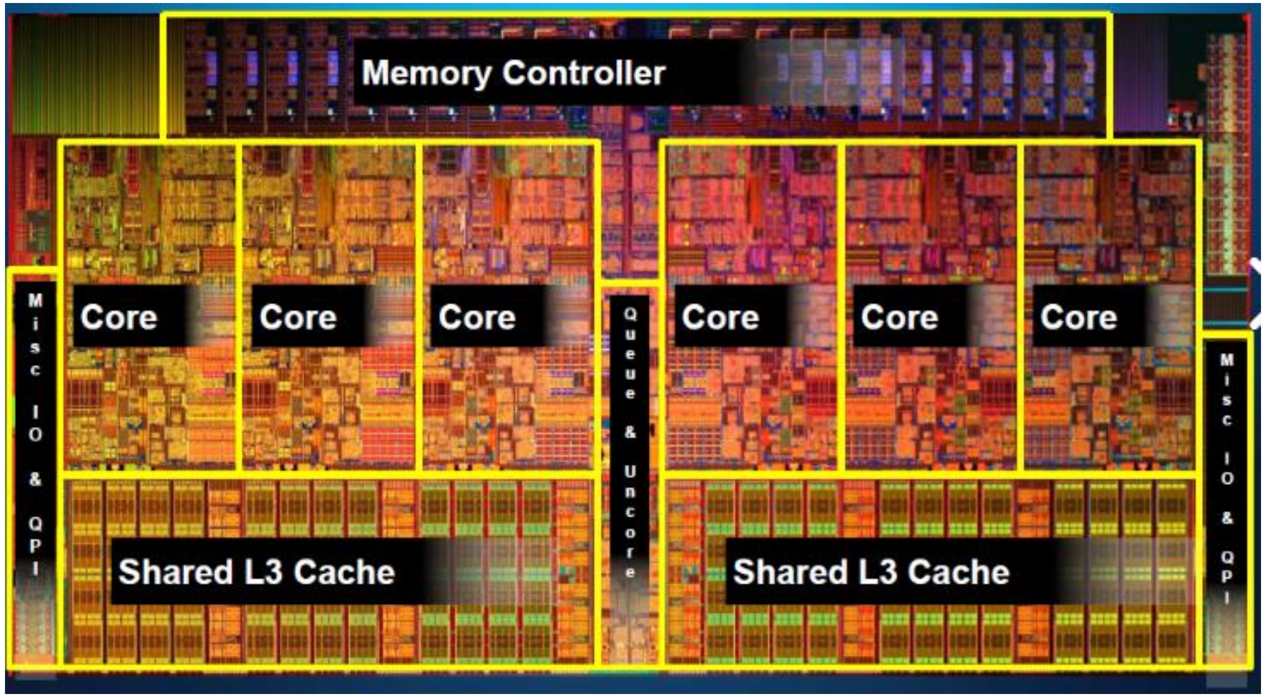


- Use smaller L1 if there is also L2
- Use simpler write-through L1 with on-chip L2

Things getting Complicated ...

- When you have multi core, memory coherence is another problem.

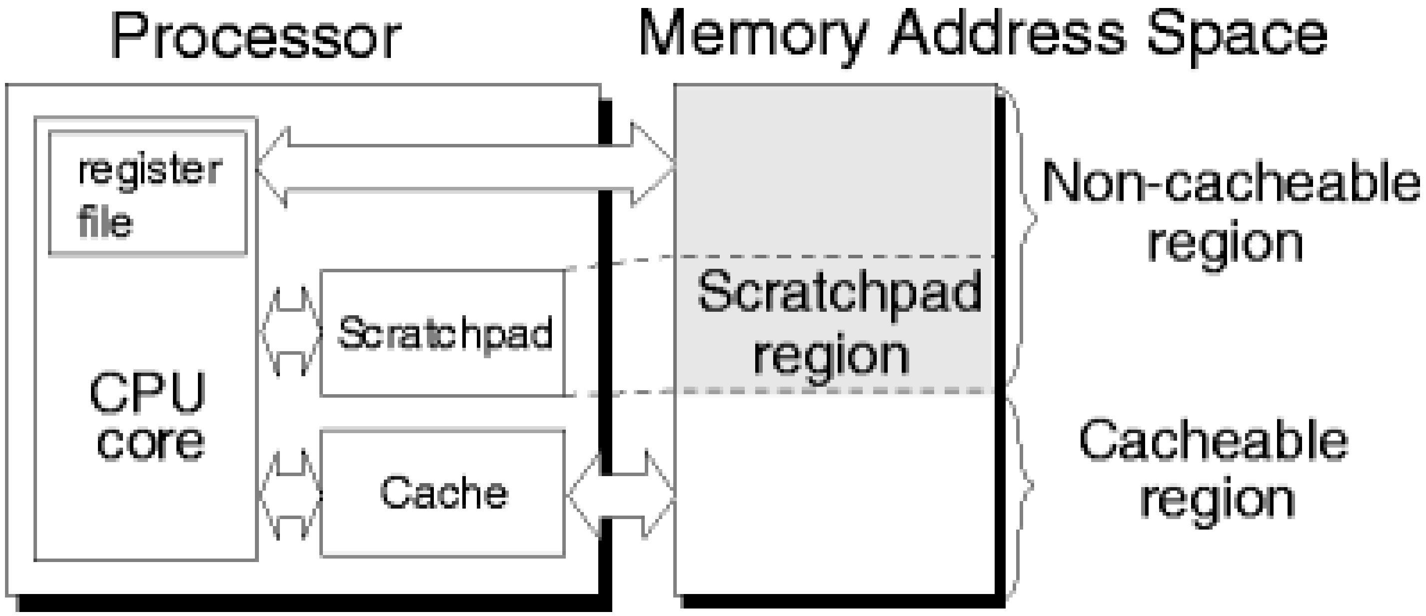
- Intel Core i7 980
  - 32KB Level 1 Data Cache
  - 32KB Level 1 Instruction Cache
  - 256KB Level 2 Cache
  - 12MB Level 3 Cache



Scratch Pad

- Cache is so complicated
- Do we really need such complicated schemes for specific applications? Ex. AI

- Scratchpad: something do not written in main memory
- Directly manipulated by applications (like GPU)



Pipelined Architecture with Caches

- Assuming we only have one kind of address: physical address → corresponding to address lines of actual hardware memory.
- Again forward logic is necessary when data is miss but ready.