CS 152 Computer Architecture and Engineering
CS252 Graduate Computer Architecture

Lecture 5 – Memory

Krste Asanovic
Electrical Engineering and Computer Sciences
University of California at Berkeley

http://www.eecs.berkeley.edu/~krste
http://inst.eecs.berkeley.edu/~cs152

Last time in Lecture 4

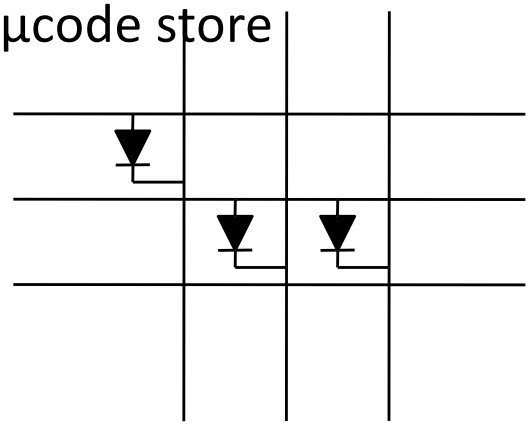
- Handling exceptions in pipelined machines by passing exceptions down pipeline until instructions cross commit point in order
- Can use values before commit through bypass network
- Pipeline hazards can be avoided through software techniques: scheduling, loop unrolling
- Decoupled architectures use queues between “access” and “execute” pipelines to tolerate long memory latency
- Regularizing all functional units to have same latency simplifies more complex pipeline design by avoiding structural hazards, can be expanded to in-order superscalar designs

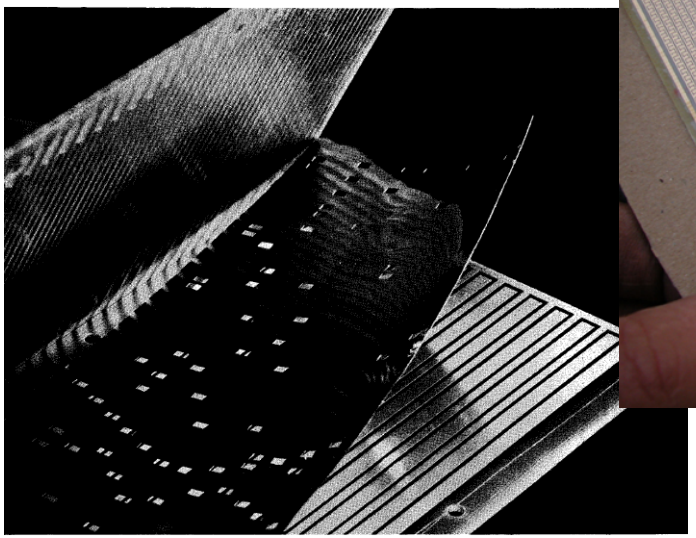
Early Read-Only Memory Technologies

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

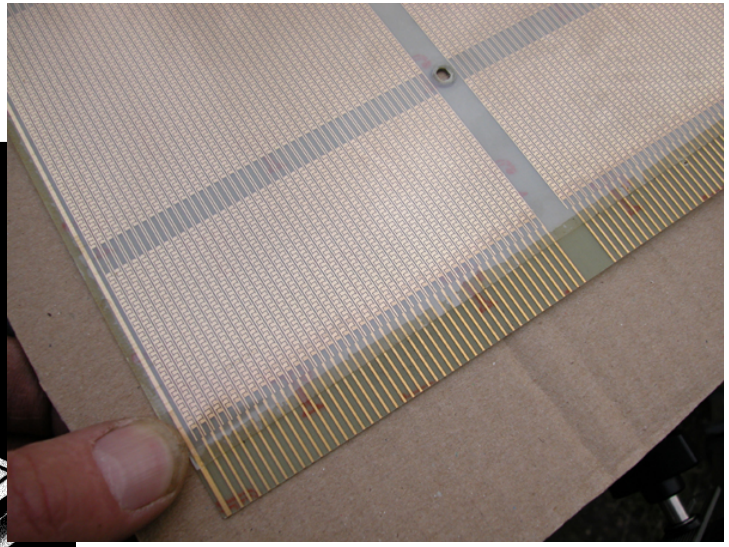
Punched paper tape, instruction stream in Harvard Mk 1

Diode Matrix, EDSAC-2 µcode store





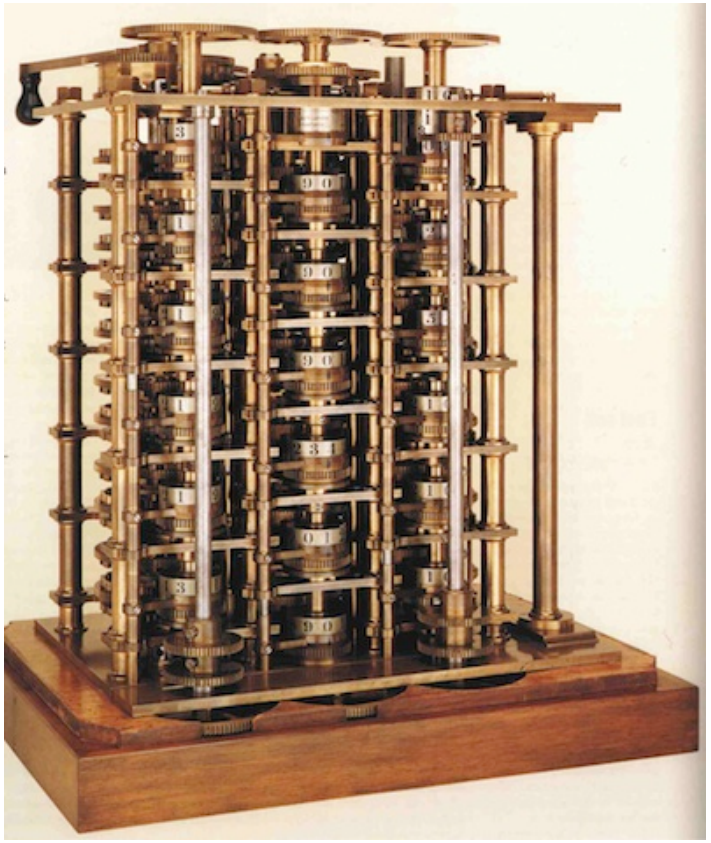
IBM Card Capacitor ROS

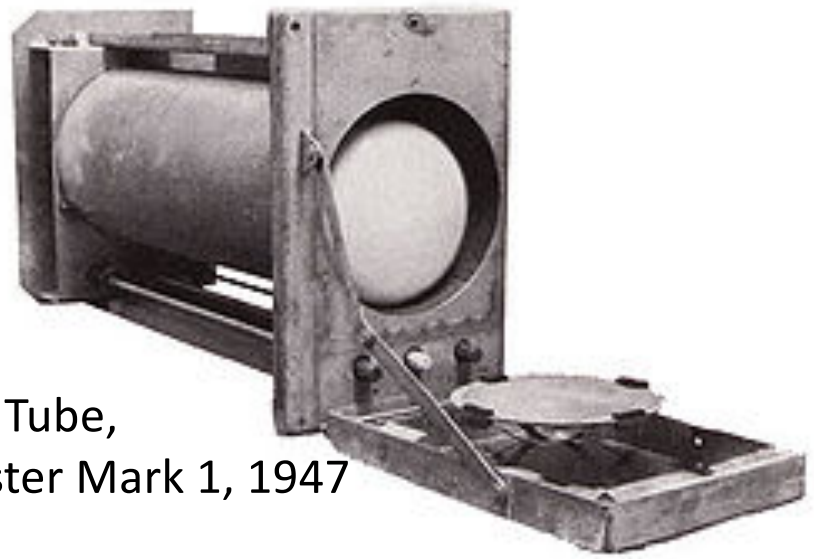


IBM Balanced Capacitor ROS

Early Read/Write Main Memory Technologies

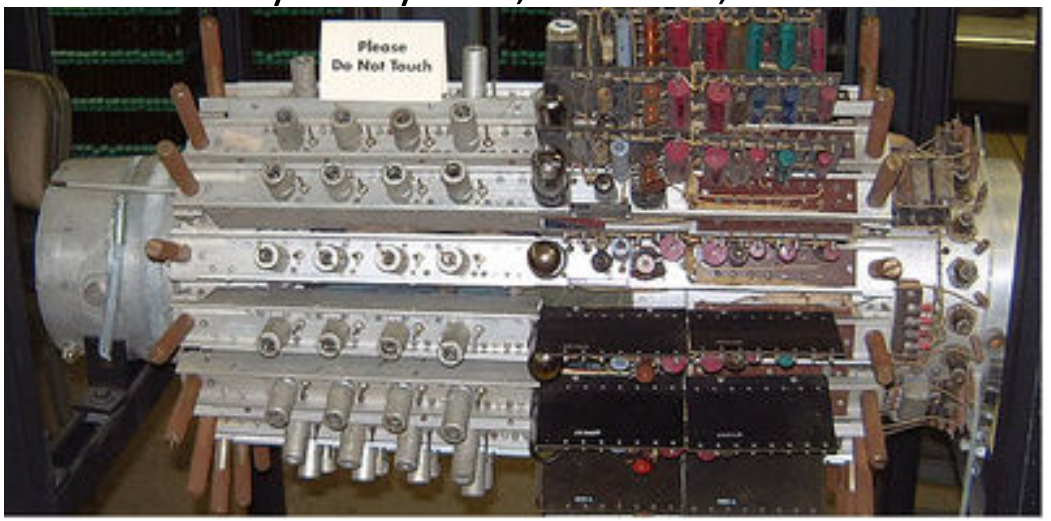
Babbage, 1800s: Digits stored on mechanical wheels





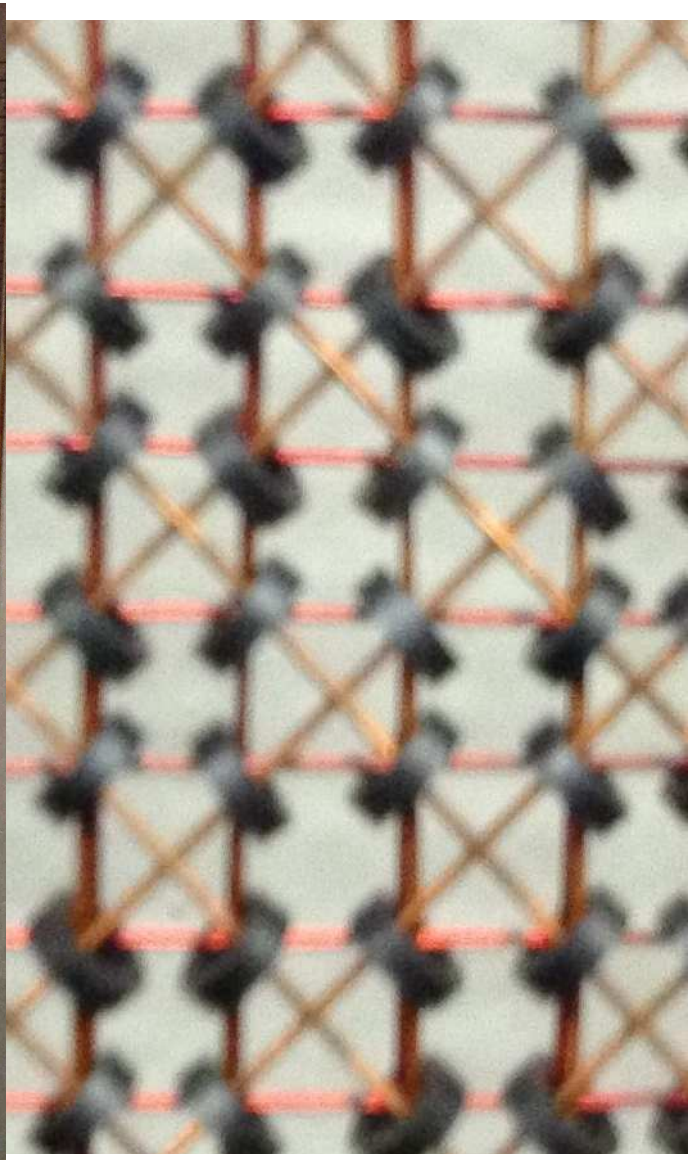
Williams Tube, Manchester Mark 1, 1947

Mercury Delay Line, Univac 1, 1951



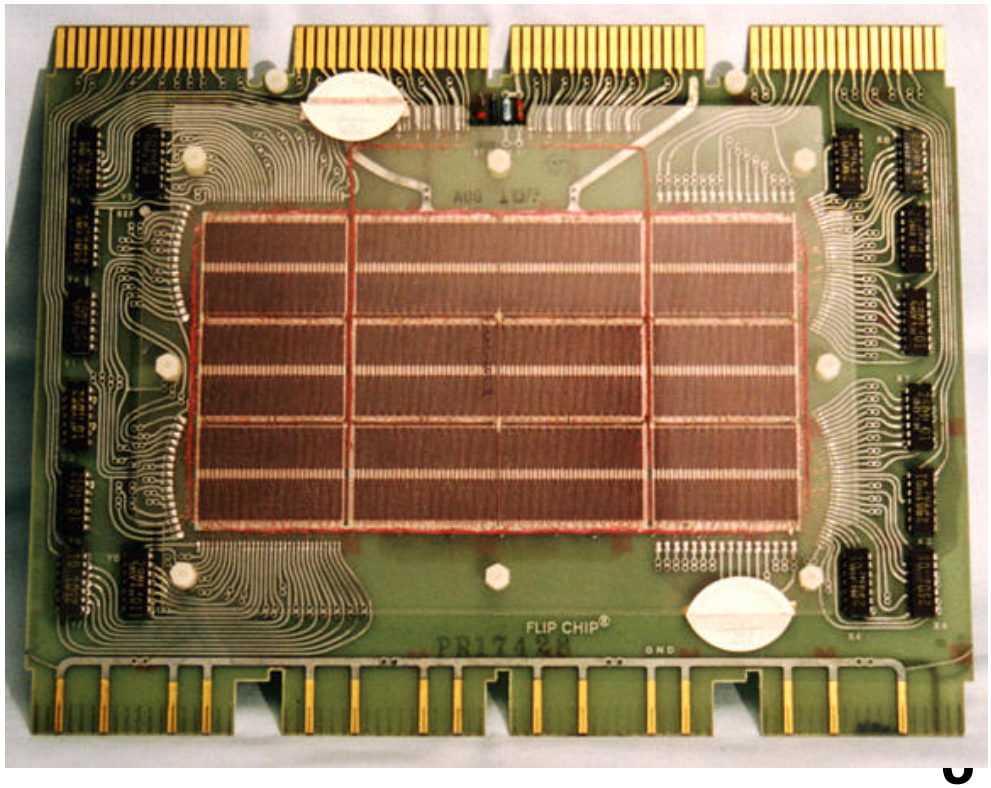
Also, regenerative capacitor memory on Atanasoff-Berry computer, and rotating magnetic drum memory on IBM 650

MIT Whirlwind Core Memory



Core Memory

- Core memory was first large scale reliable main memory
  - invented by Forrester in late 40s/early 50s at MIT for Whirlwind project
- Bits stored as magnetization polarity on small ferrite cores threaded onto two-dimensional grid of wires
- Coincident current pulses on X and Y wires would write cell and also sense original state (destructive reads)
- Robust, non-volatile storage
- Used on space shuttle computers
- Cores threaded onto wires by hand (25 billion a year at peak production)
- Core access time ~ 1µs

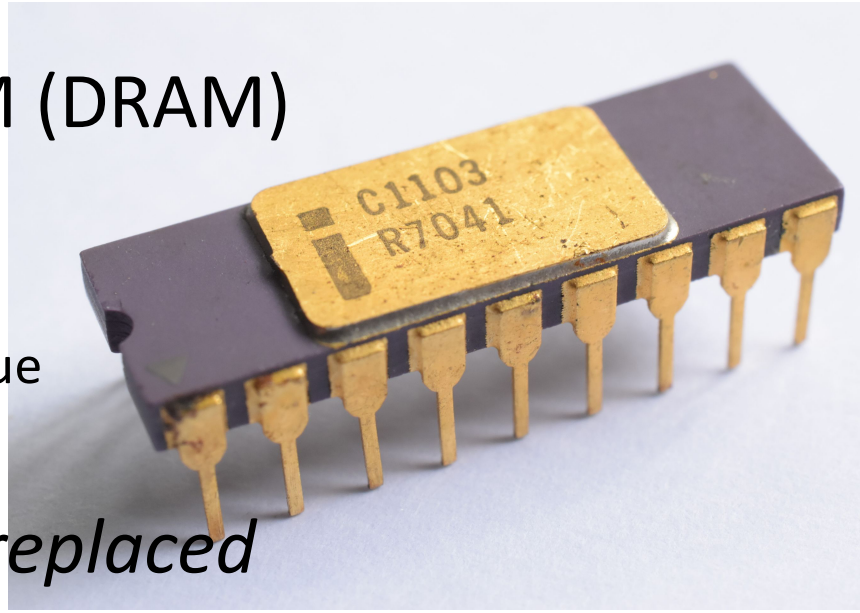


DEC PDP-8/E Board,
4K words x 12 bits, (1968)

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). SRAM cell internals similar to a latch (cross-coupled inverters).

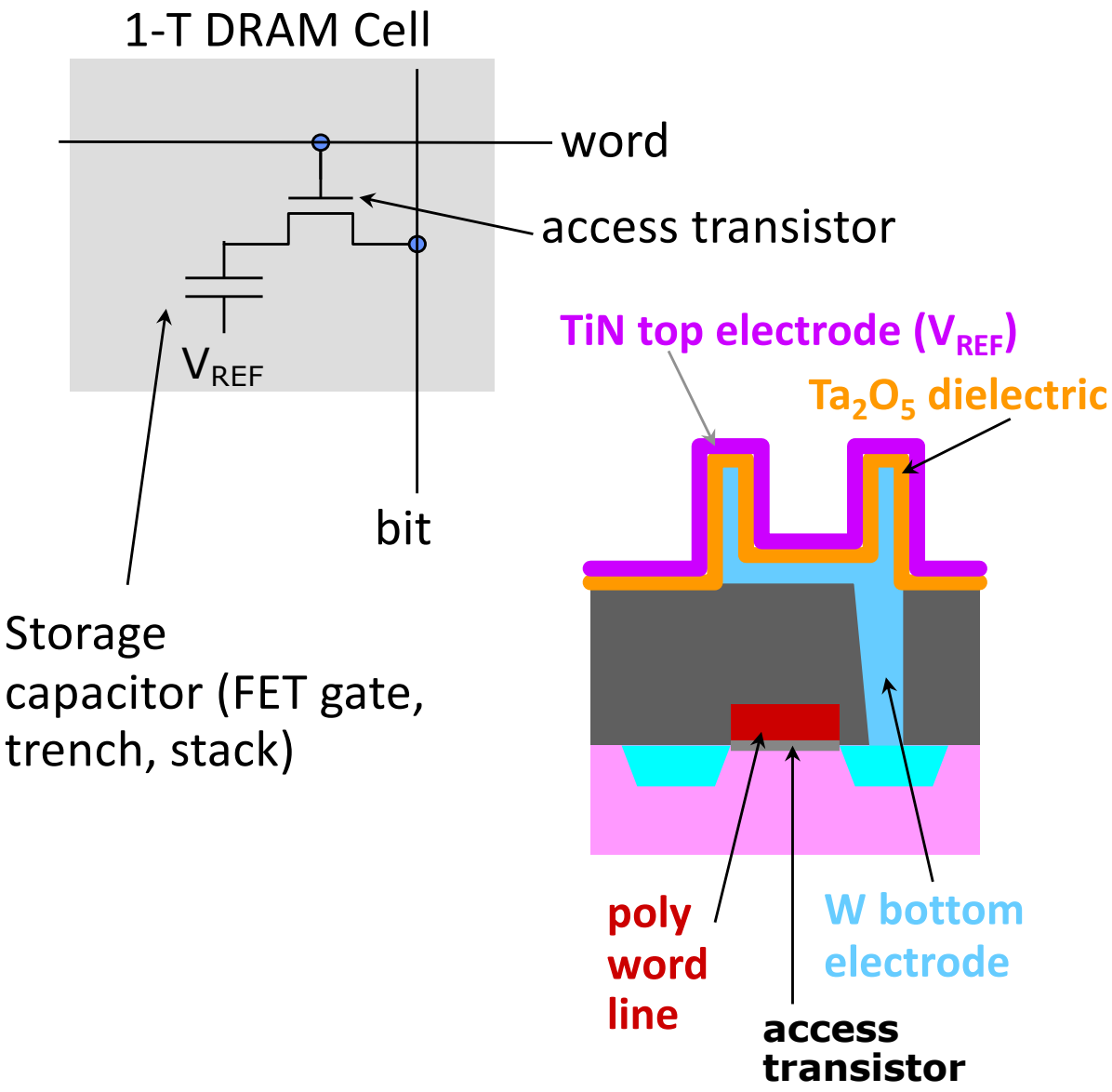
- First commercial Dynamic RAM (DRAM) was Intel 1103
  - 1Kbit of storage on single chip
  - charge on a capacitor used to hold value

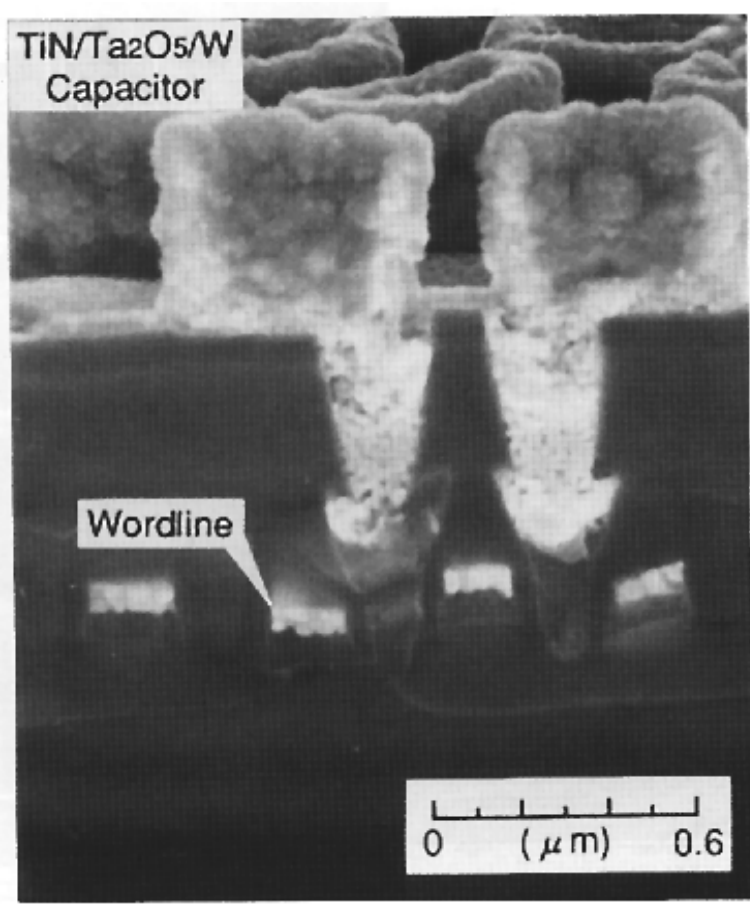


[Thomas Nguyen CC-BY-SA]

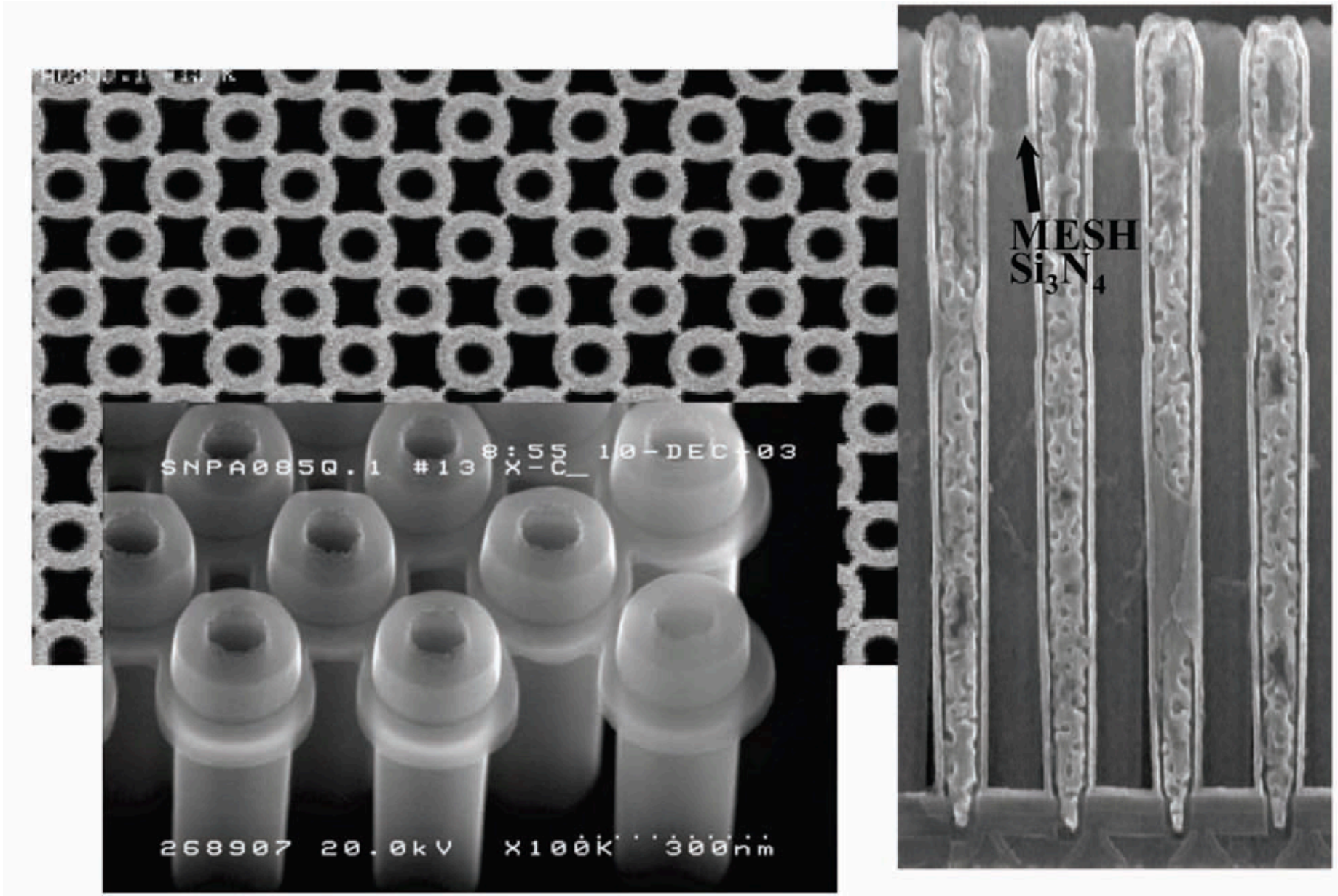
Semiconductor memory quickly replaced core in '70s

One-Transistor Dynamic RAM [Dennard, IBM]



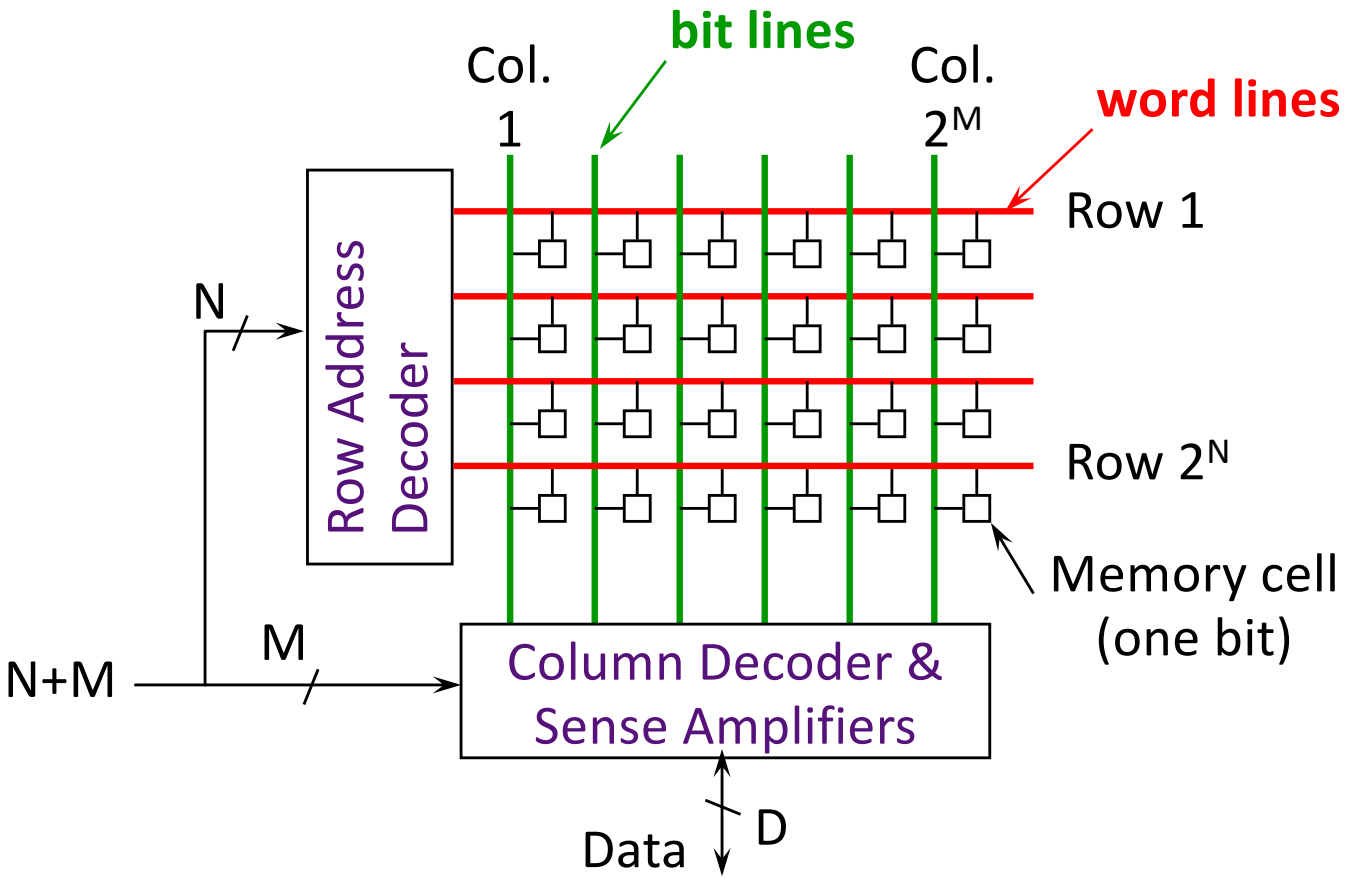


Modern DRAM Structure



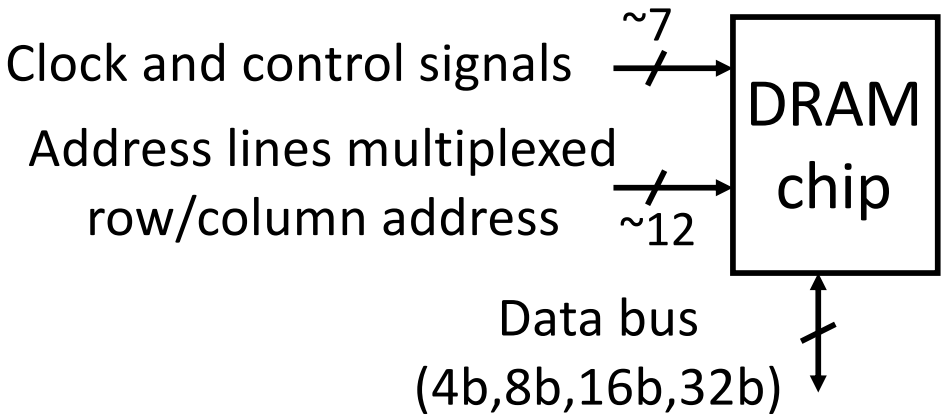
[Samsung, sub-70nm DRAM, 2004]

DRAM Architecture

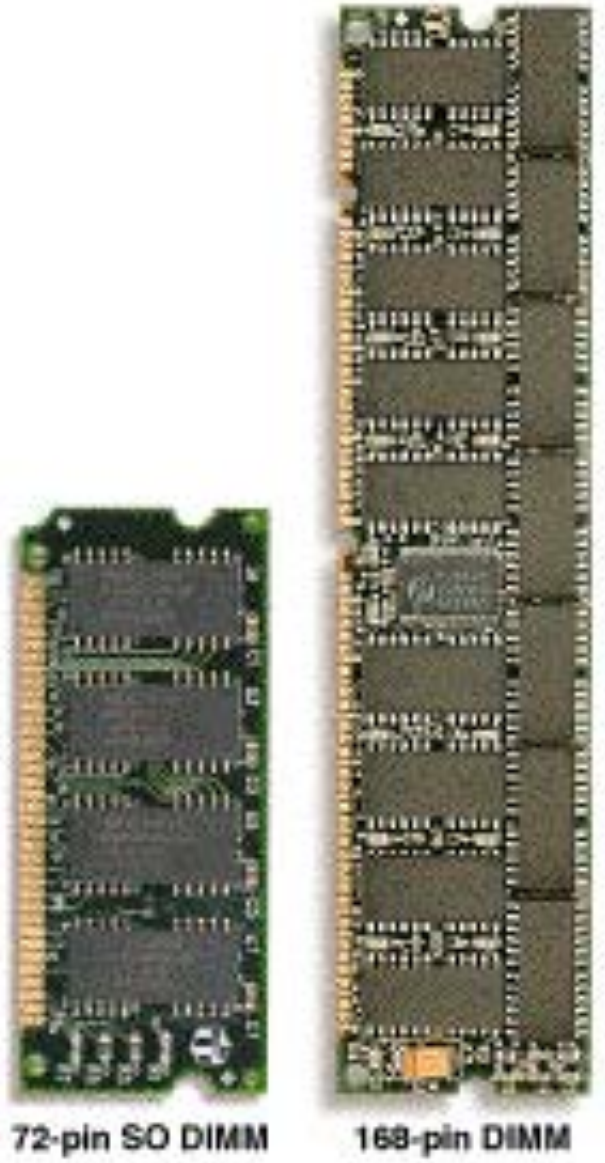


- Bits stored in 2-dimensional arrays on chip
- Modern chips have around 4-8 logical banks on each chip
  - each logical bank physically implemented as many smaller arrays

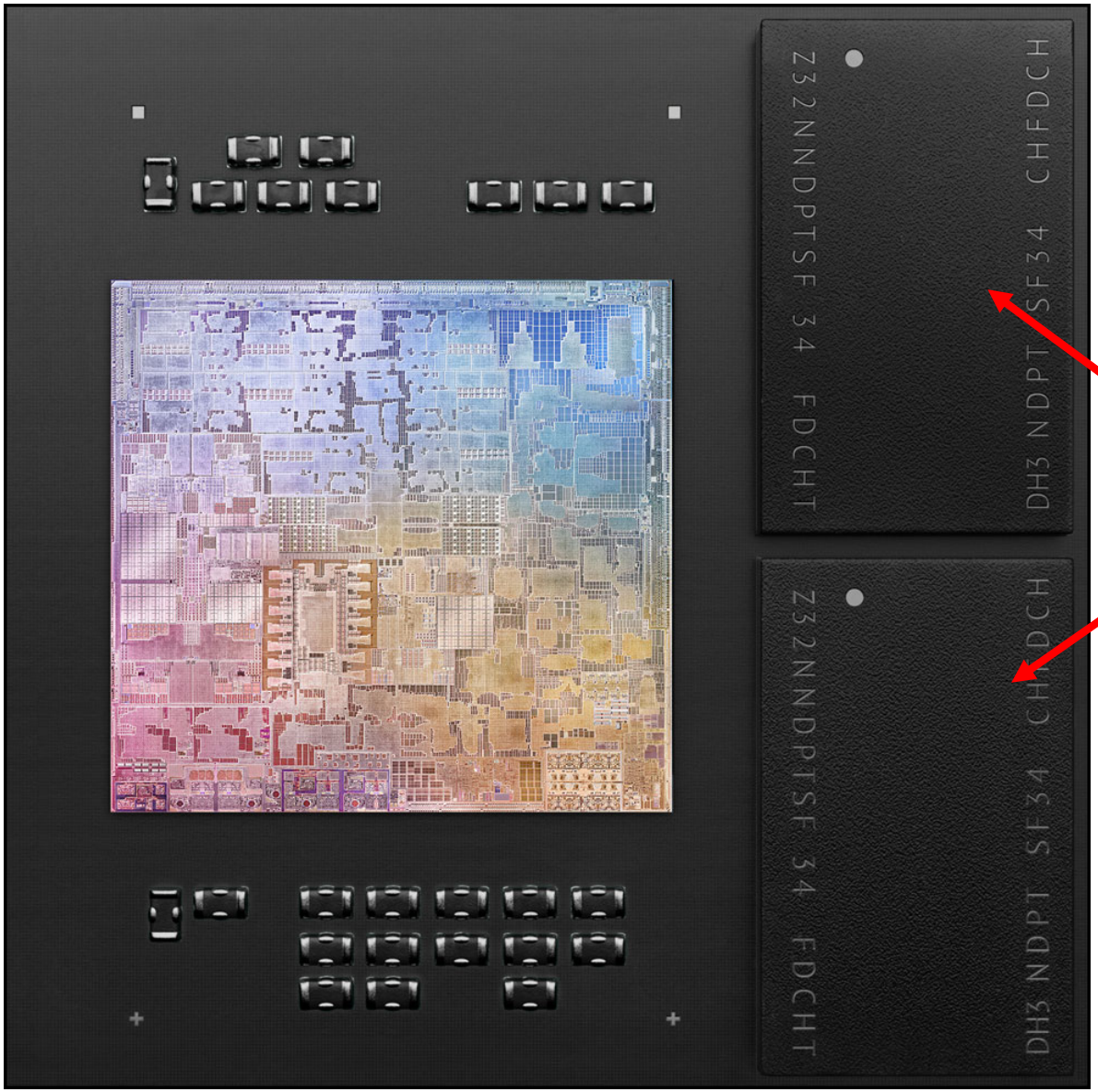
DRAM Packaging
(Laptops/Desktops/Servers)



- 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)



DRAM Packaging, Apple M1



Two DRAM chips on same package as system SoC

- 128b databus, running at 4.2Gb/s
- 68GB/s bandwidth

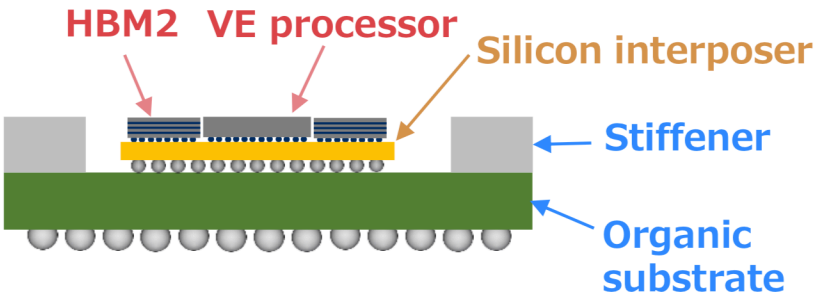
High-Bandwidth Memory in SX-Aurora

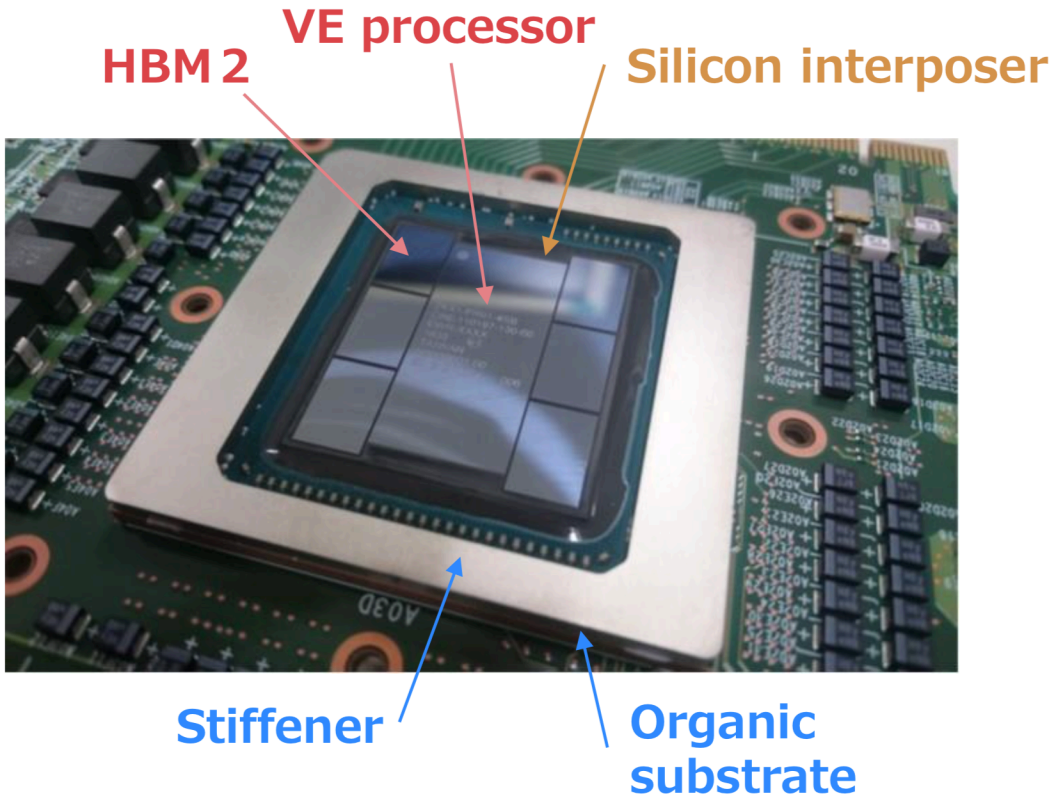
Vector Engine Processor Module

SX-Aurora TSUBAS

2.5D implementation

- A VE processor and six 8Hi or 4Hi HBM2 modules on a silicon interposer
- Lidless package to minimize thermal resistance
- Package size: 60mm x 60mm
- Interposer size: 32.5mm x 38mm
- VE processor size: 15mm x 33mm





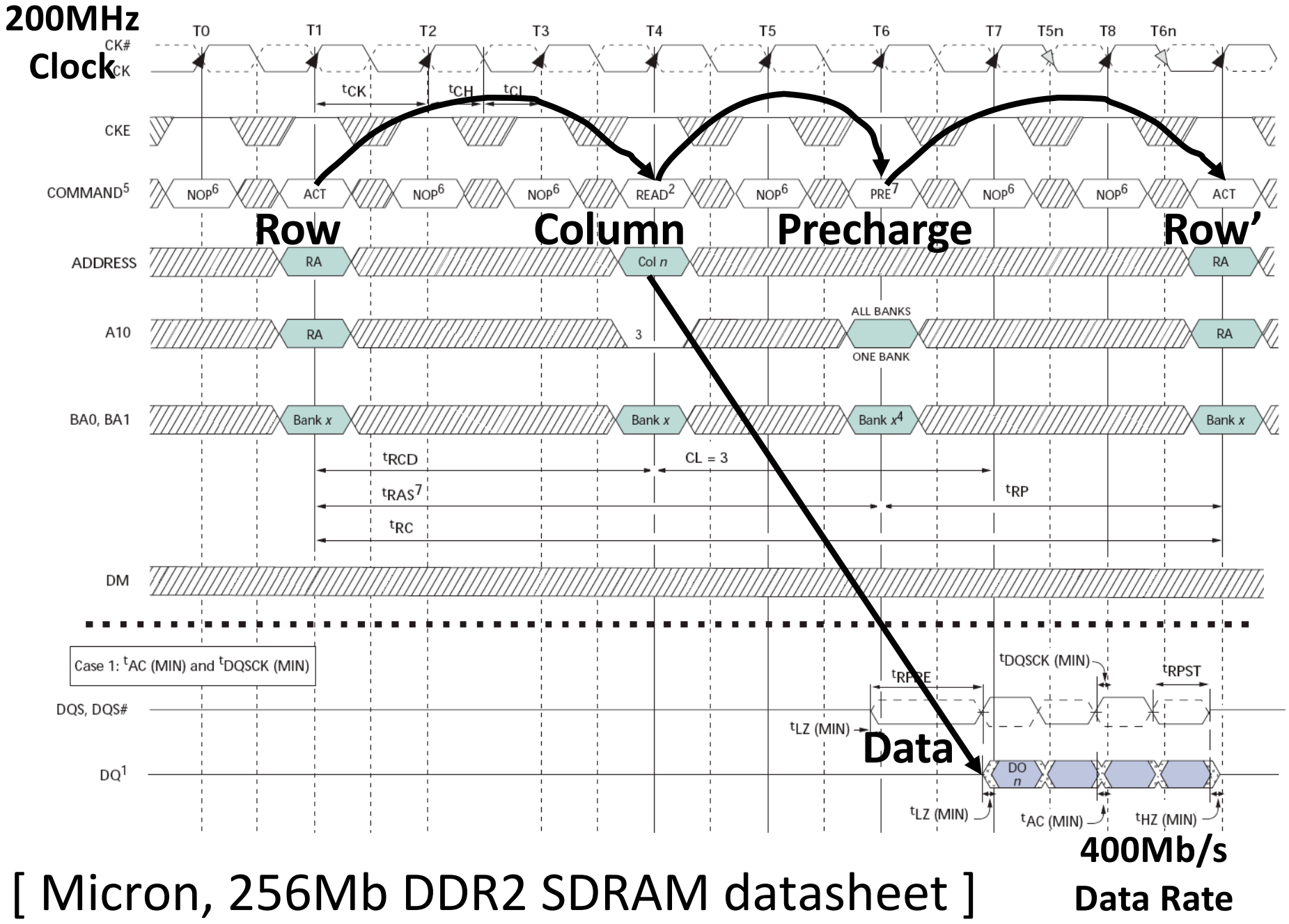
World's first implementation of a processor with 6 HBM2s

*1.2TB/s HBM Bandwidth, 6 channels * 1024 bits/channel * 1.6 Gb/s/pin*

DRAM Operation

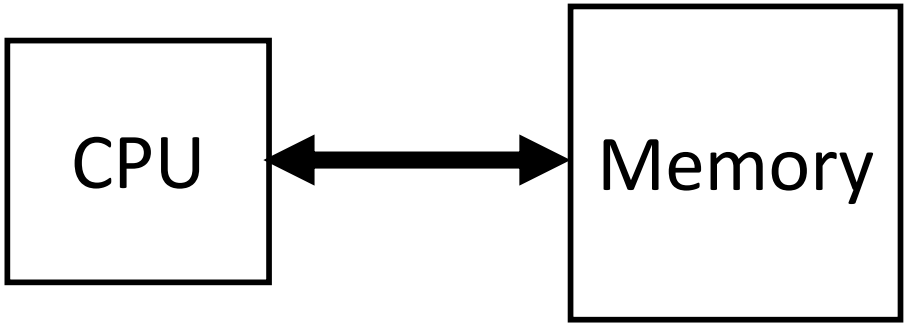
- Three steps in read/write access to a given bank
- Row access (RAS)
  - decode row address, enable addressed row (often multiple Kb in row)
  - bitlines share charge with storage cell
  - small change in voltage detected by sense amplifiers which latch whole row of bits
  - sense amplifiers drive bitlines full rail to recharge storage cells
- Column access (CAS)
  - decode column address to select small number of sense amplifier latches (4, 8, 16, or 32 bits depending on DRAM package)
  - on read, send latched bits out to chip pins
  - on write, change sense amplifier latches which then charge storage cells to required value
  - can perform multiple column accesses on same row without another row access (burst mode)
- Precharge
  - charges bit lines to known value, required before next row access
- Each step has a latency of around 15-20ns in modern DRAMs
- Various DRAM standards (DDR, RDRAM) have different ways of encoding the signals for transmission to the DRAM, but all share same core architecture

Double-Data Rate (DDR2) DRAM



[Micron, 256Mb DDR2 SDRAM datasheet]

CPU-Memory Bottleneck



Performance of high-speed computers is usually limited by memory bandwidth & latency

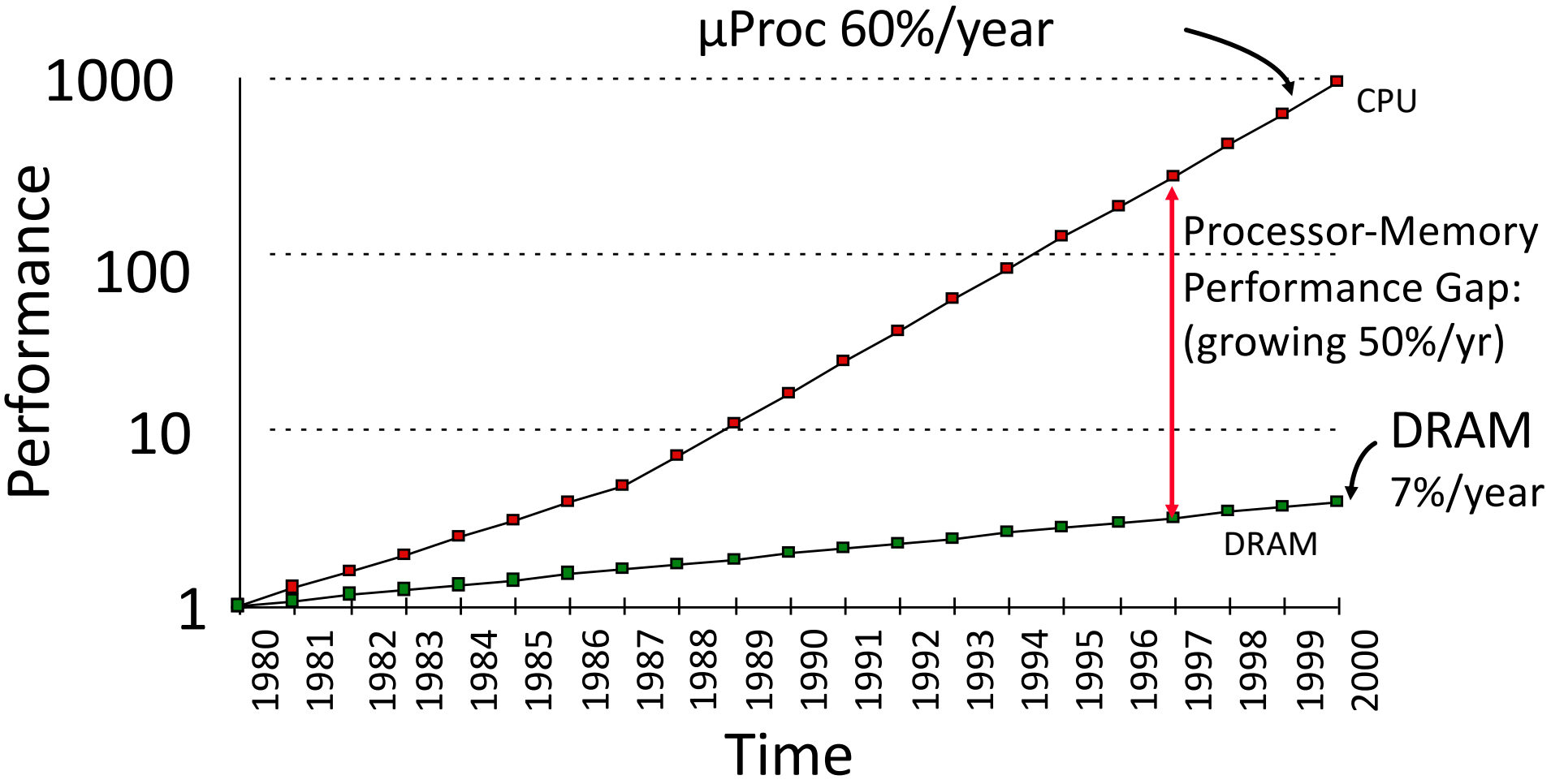
- Latency (time for a single access)
  - Memory access time >> Processor cycle time
- Bandwidth (number of accesses per unit time)

  if fraction m of instructions access memory

  ⇒ 1+m memory references / instruction

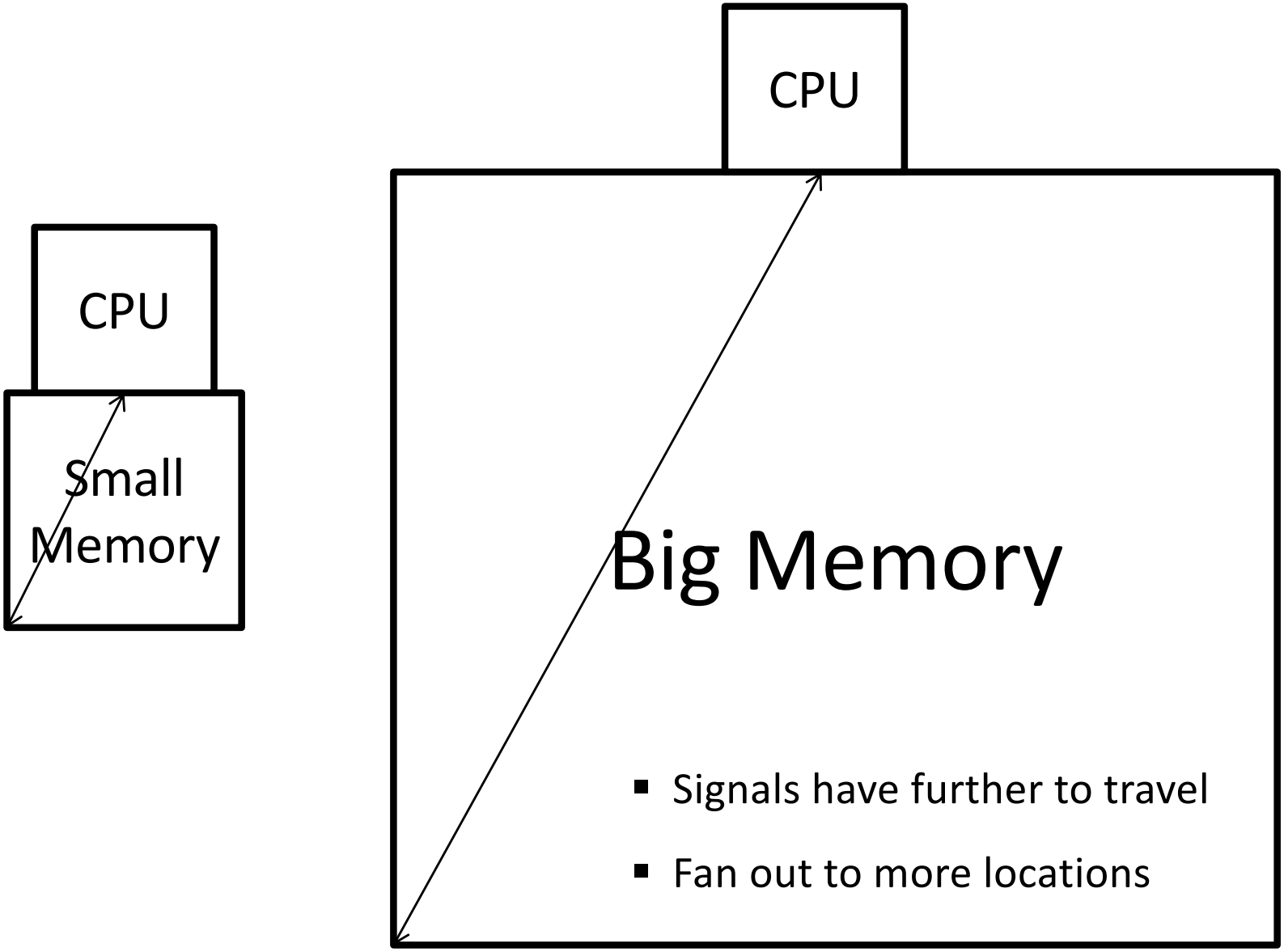
  ⇒ CPI = 1 requires 1+m memory refs / cycle (assuming RISC-V ISA)
- *Also, Occupancy (time a memory bank is busy with one request)*

Processor-DRAM Gap (latency)



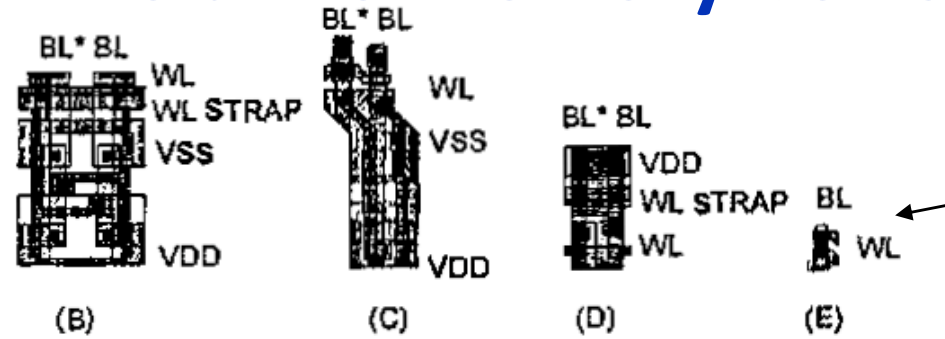
Four-issue 3GHz superscalar accessing 100ns DRAM could execute 1,200 instructions during time for one memory access!

Physical Size Affects Latency

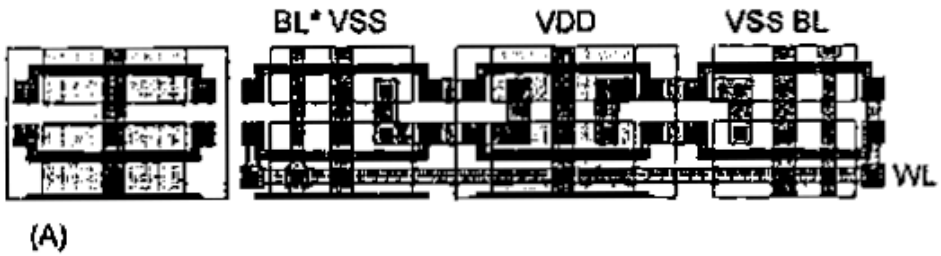


Relative Memory Cell Sizes

On-Chip SRAM in logic chip



DRAM on memory chip



1 Memory cell in 0.5µm processes

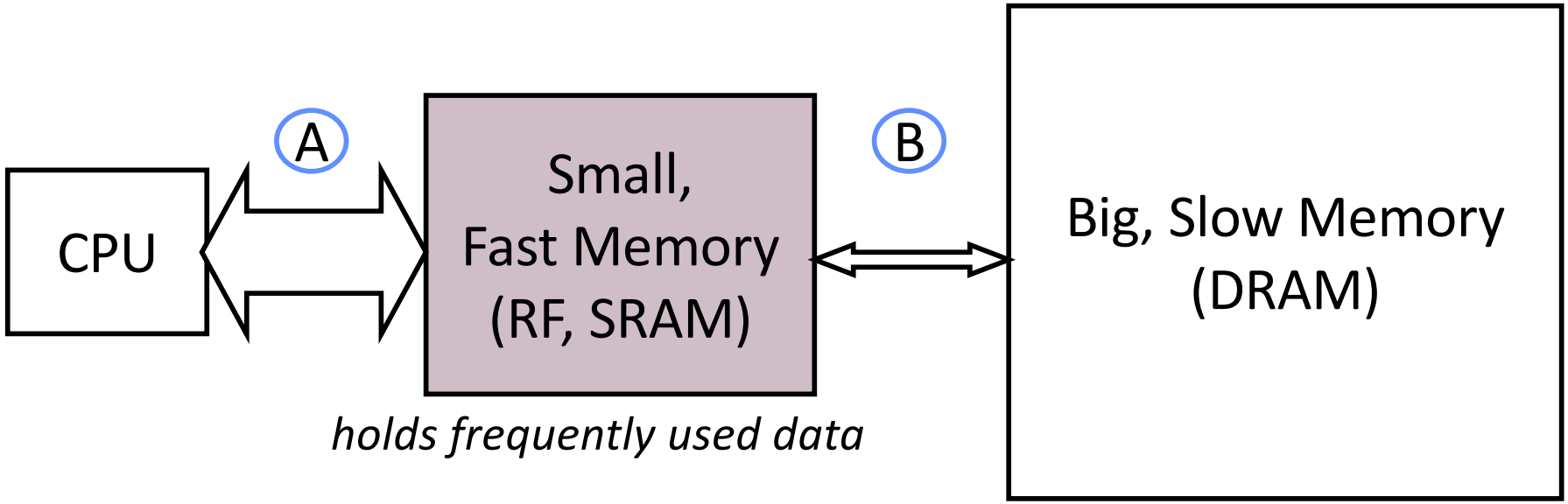
a) Gate Array SRAM
b) Embedded SRAM
c) Standard SRAM (6T cell with local interconnect)
d) ASIC DRAM
e) Standard DRAM (stacked cell)

[Foss, "Implementing Application-Specific Memory", ISSCC 1996]

Memory	Process	Cell size (μm^2)	Cell efficiency	Bits in $100mm^2 (10^3)$	Gate size (μm^2)	Gate utilization	Gates in $100mm^2 (10^3)$
Gate array SRAM	3-metal ASIC	370	80%	216	185	70%	378
Embedded SRAM	3-metal ASIC	67	70%	1045	185	70%	378
Standard SRAM	2-metal 6T local int.	43	65%	1512	245	40%	163
Embedded ASIC-DRAM	3-metal ASIC	23	60%	2609	185	70%	378
Standard DRAM	2-metal stacked cell	3.2	50%	15625	411	40%	97

Table 1: Memory and logic density for a variety of 0.5µm implementations.

Memory Hierarchy



- *capacity*: Register << SRAM << DRAM
- *latency*: Register << SRAM << DRAM
- *bandwidth:* on-chip >> off-chip

On a data access:

if data $\in$ fast memory $\Rightarrow$ low latency access *(SRAM)*

if data $\notin$ fast memory $\Rightarrow$ high latency access *(DRAM)*

CS152 Administrivia

- PS 1 due 11:59PM on Monday Feb 8
- Lab 1 due 11:59PM Wed Feb 17

CS252 Administrivia

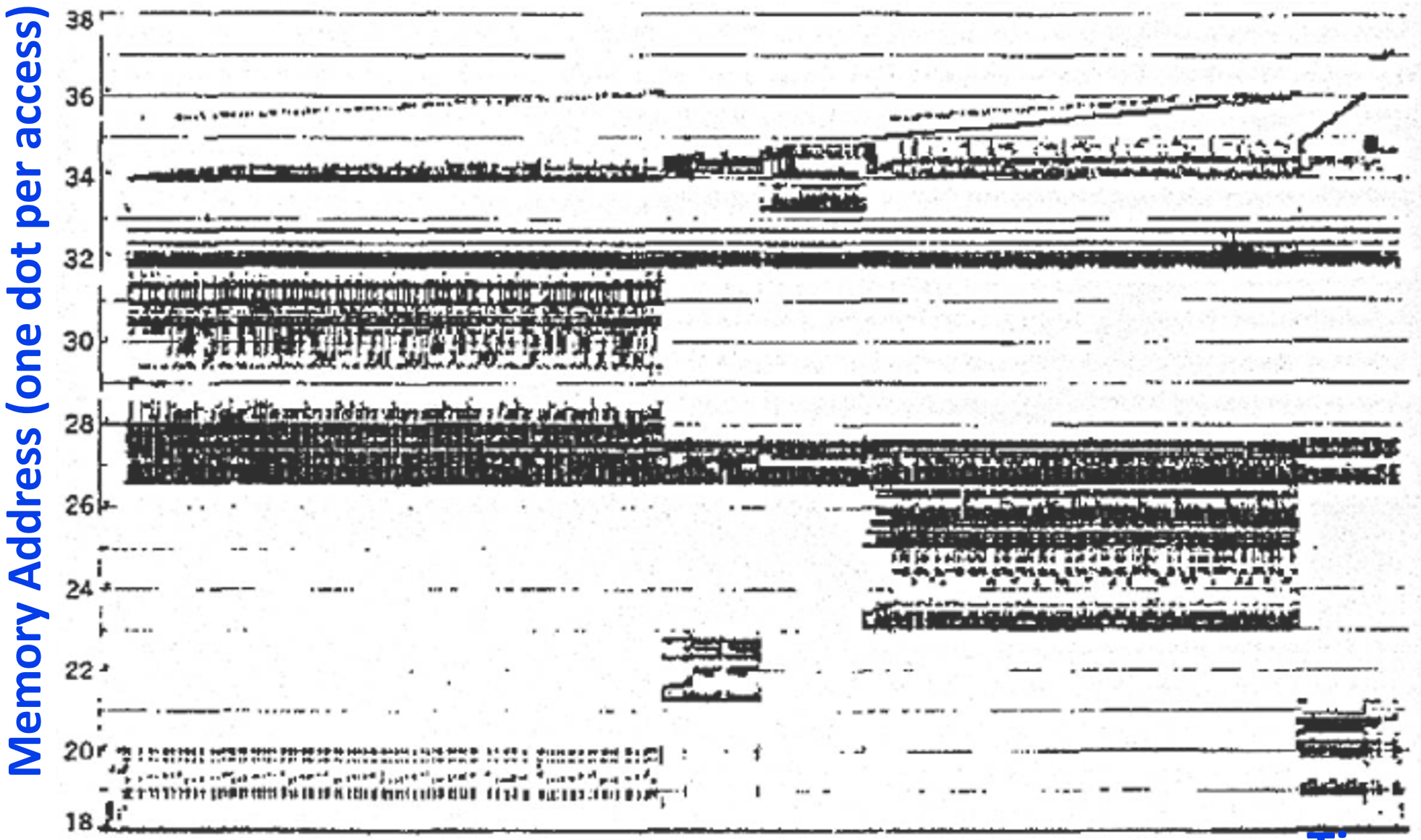
- Project proposals due 11:59PM Wed Feb 26th
- Use Krste's office hours Tue 10-11am to get feedback on ideas
  - email for link
- Readings discussion will be Thursdays 5-6pm
  - zoom link on Piazza
  - Questions on Piazza

Management of Memory Hierarchy

- Small/fast storage, e.g., registers
  - Address usually specified in instruction
  - Generally implemented directly as a register file
    - *but hardware might do things behind software's back, e.g., stack management, register renaming*

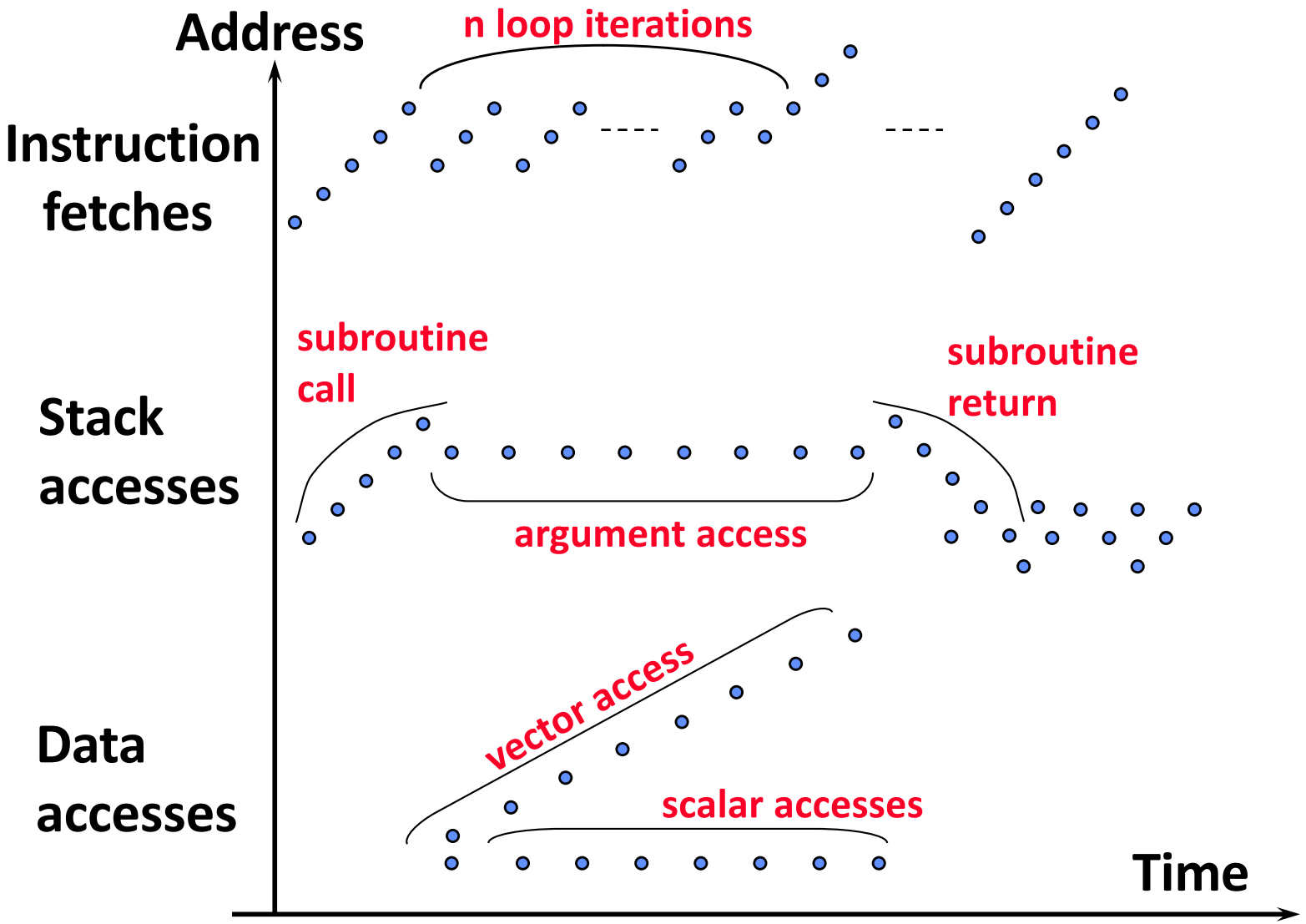
- Larger/slower storage, e.g., main memory
  - Address usually computed from values in register
  - Generally implemented as a hardware-managed cache hierarchy (hardware decides what is kept in fast memory)
    - *but software may provide "hints", e.g., don't cache or prefetch*

Real Memory Reference Patterns



Donald J. Hatfield, Jeanette Gerald: Program Restructuring for Virtual Memory. IBM Systems Journal 10(3): 168-192 (1971)

Typical Memory Reference Patterns

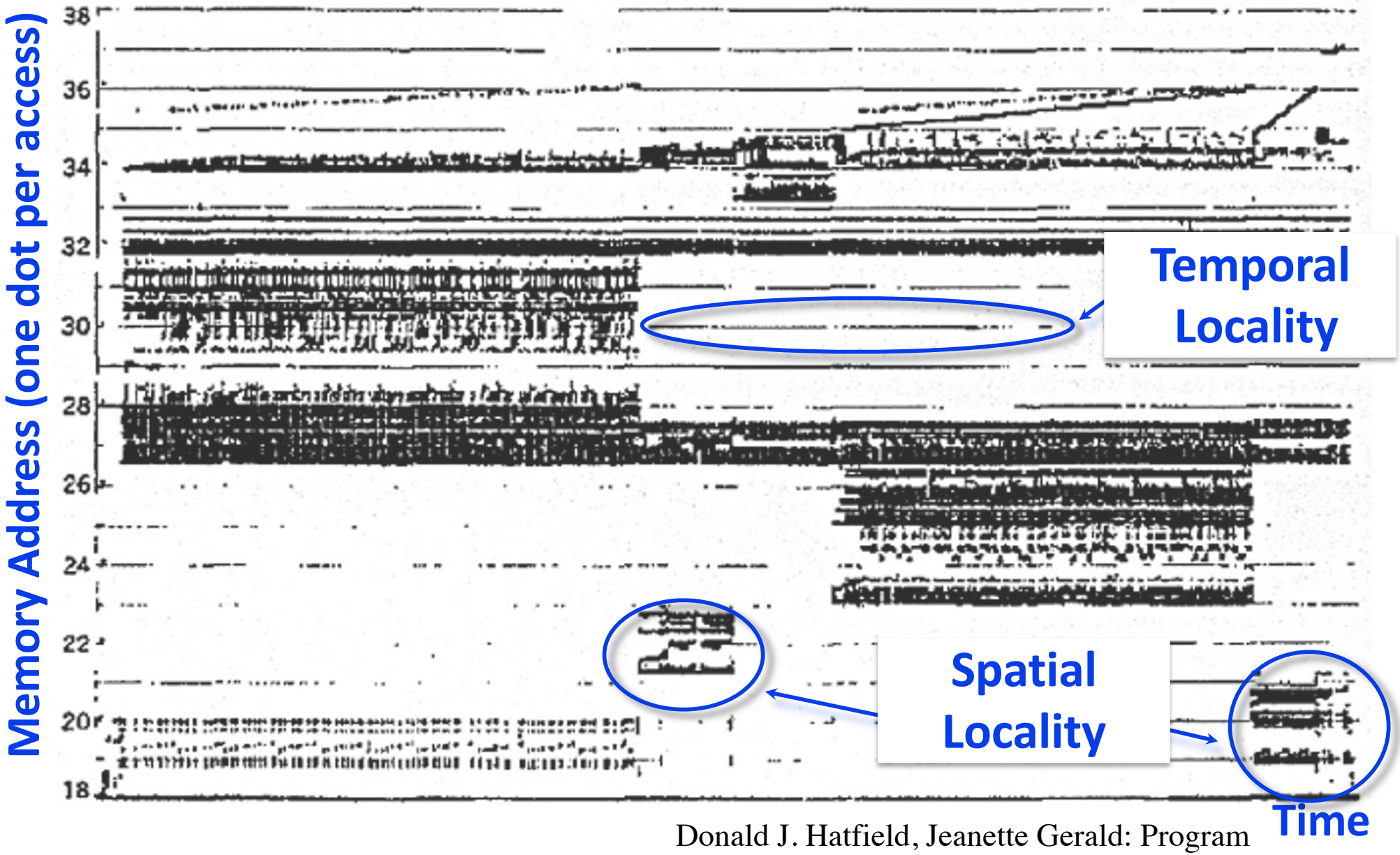


Two predictable properties of memory references:

- **Temporal Locality**: If a location is referenced it is likely to be referenced again in the near future.

- **Spatial Locality**: If a location is referenced it is likely that locations near it will be referenced in the near future.

Memory Reference Patterns



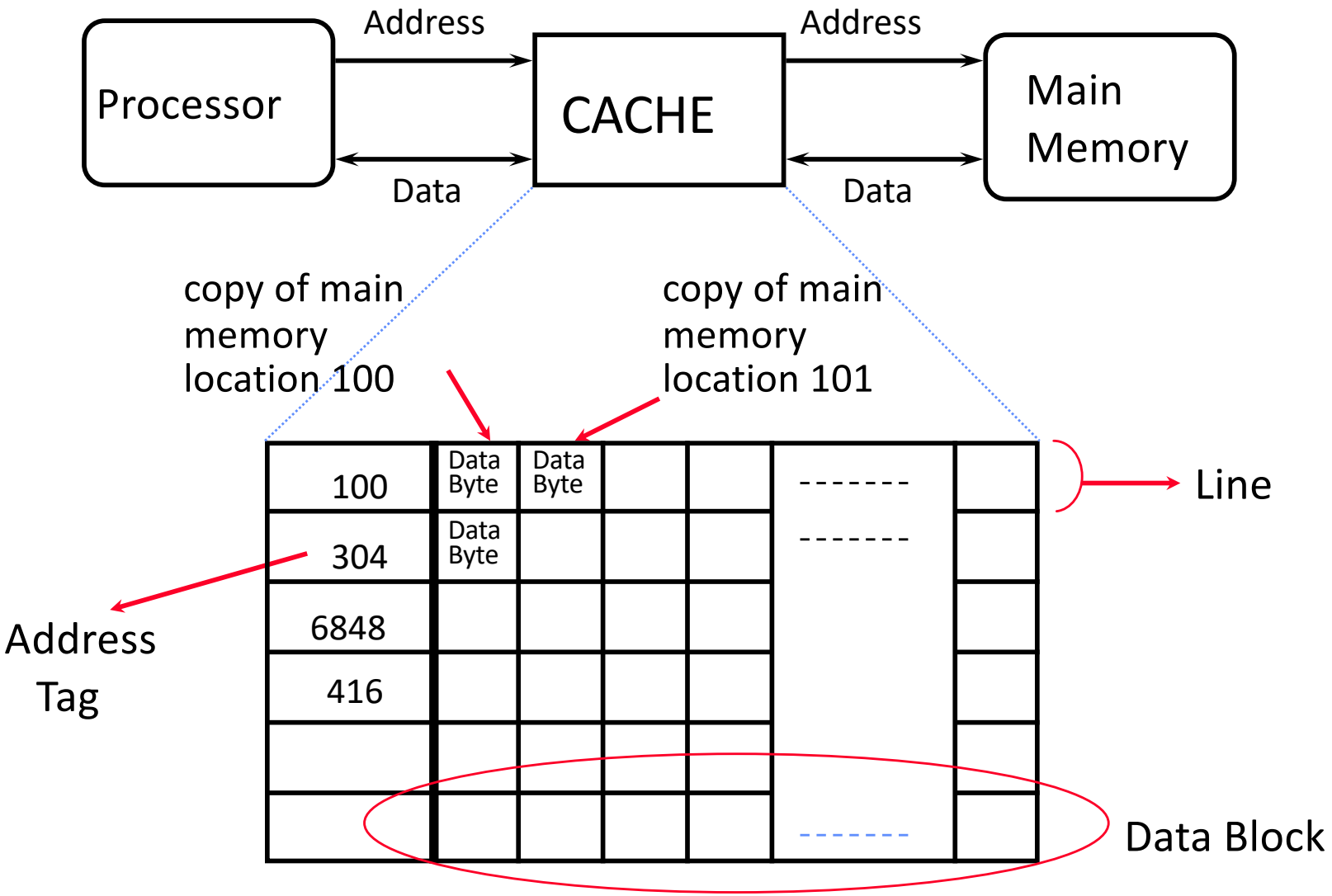
Donald J. Hatfield, Jeanette Gerald: Program Restructuring for Virtual Memory. IBM Systems Journal 10(3): 168-192 (1971)

Caches exploit both types of predictability:

- Exploit temporal locality by remembering the contents of recently accessed locations.

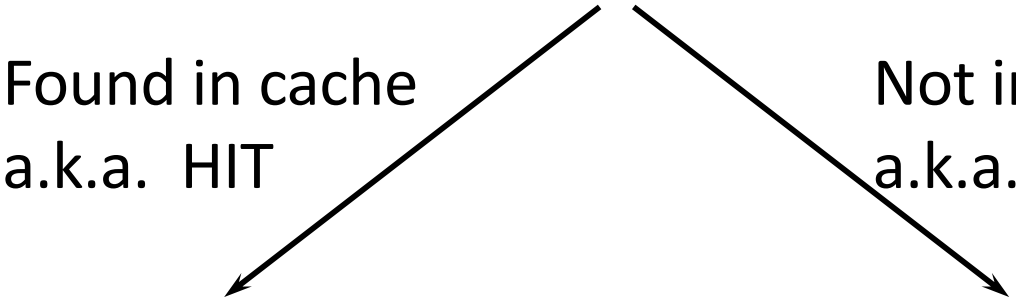
- Exploit spatial locality by fetching blocks of data around recently accessed locations.

Inside a Cache



Cache Algorithm (Read)

Look at Processor Address, search cache tags to find match. Then either



Return copy of data from cache

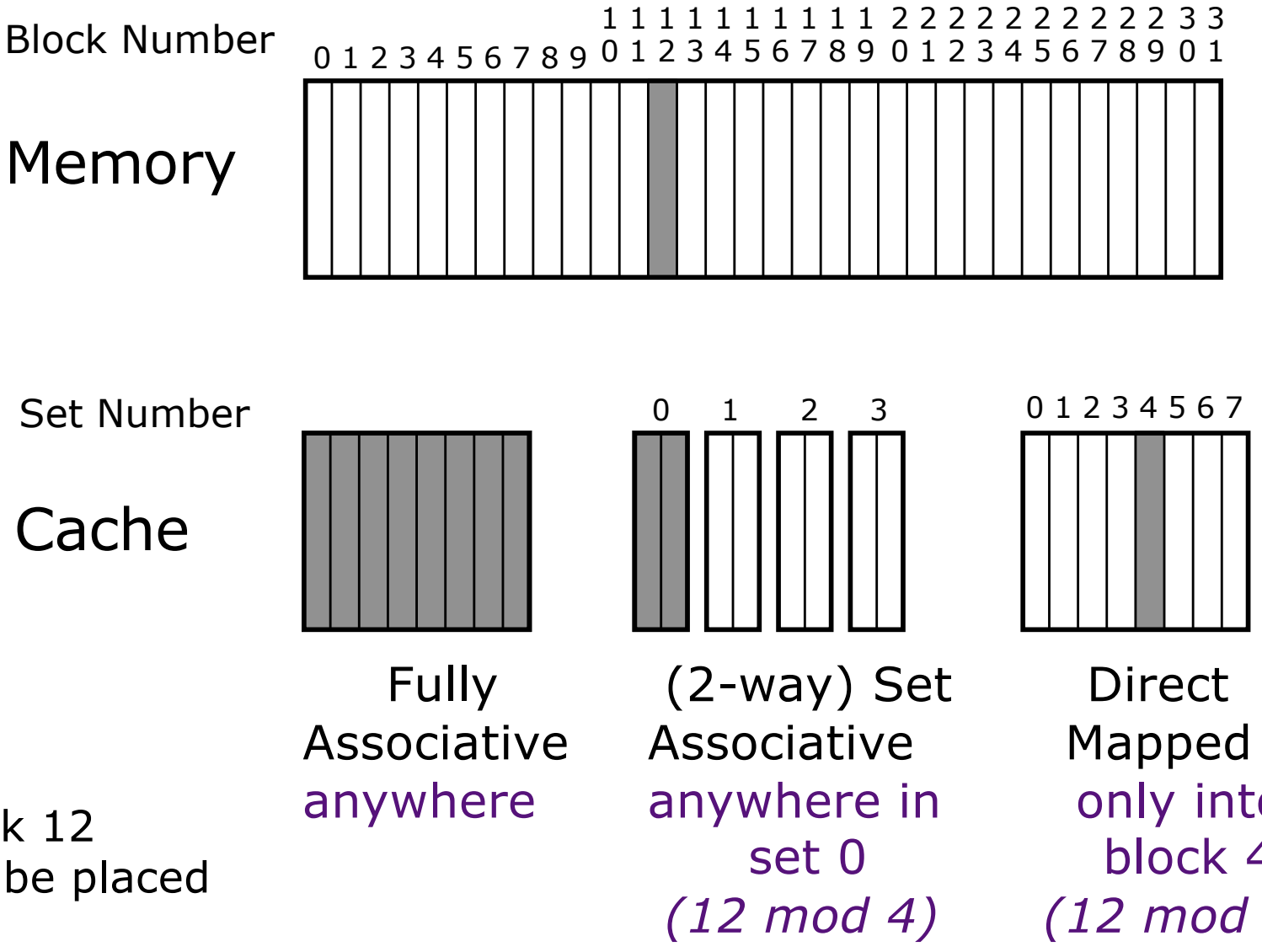
Read block of data from Main Memory

Wait …

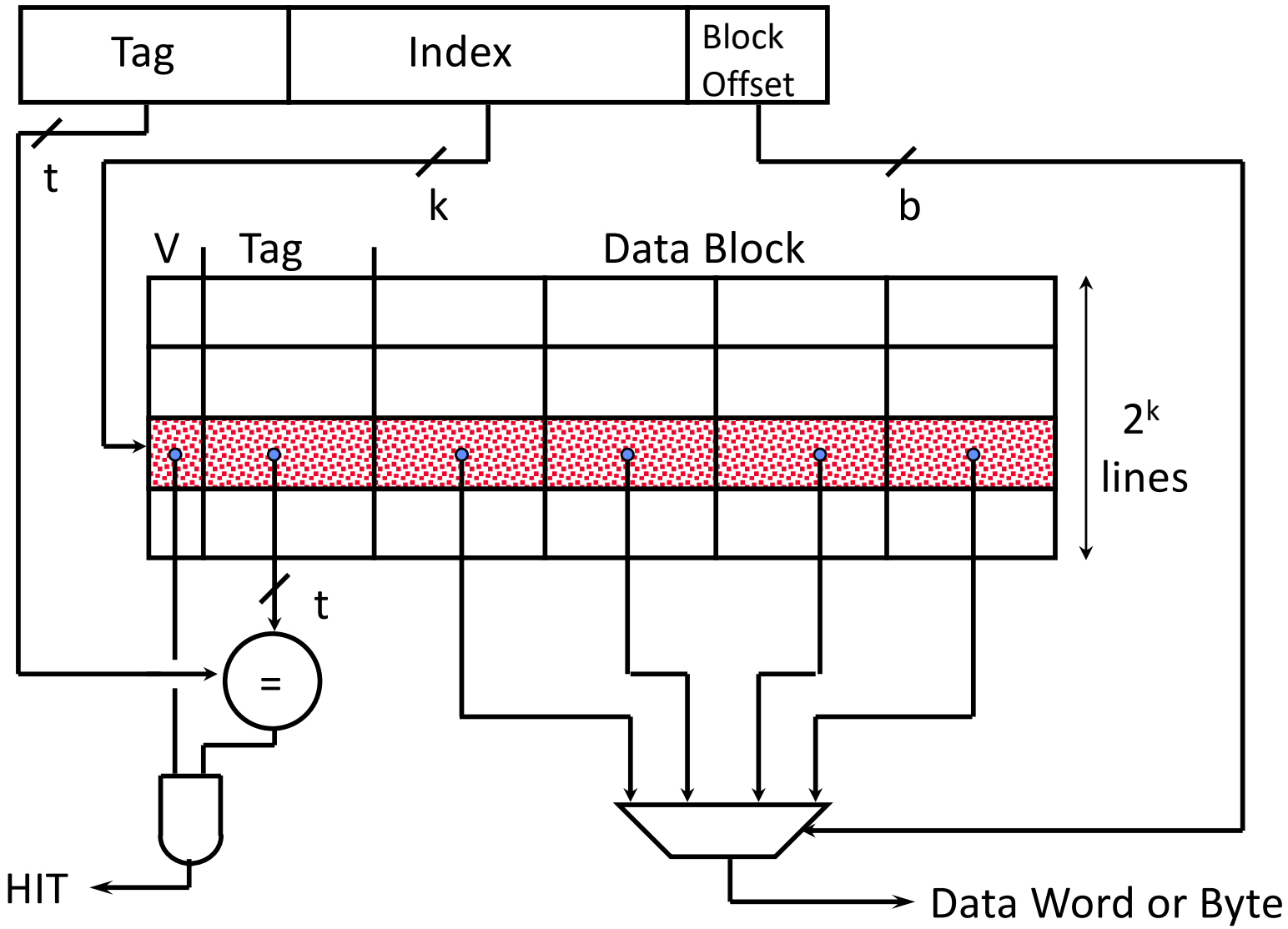
Return data to processor and update cache

Q: Which line do we replace?

Placement Policy

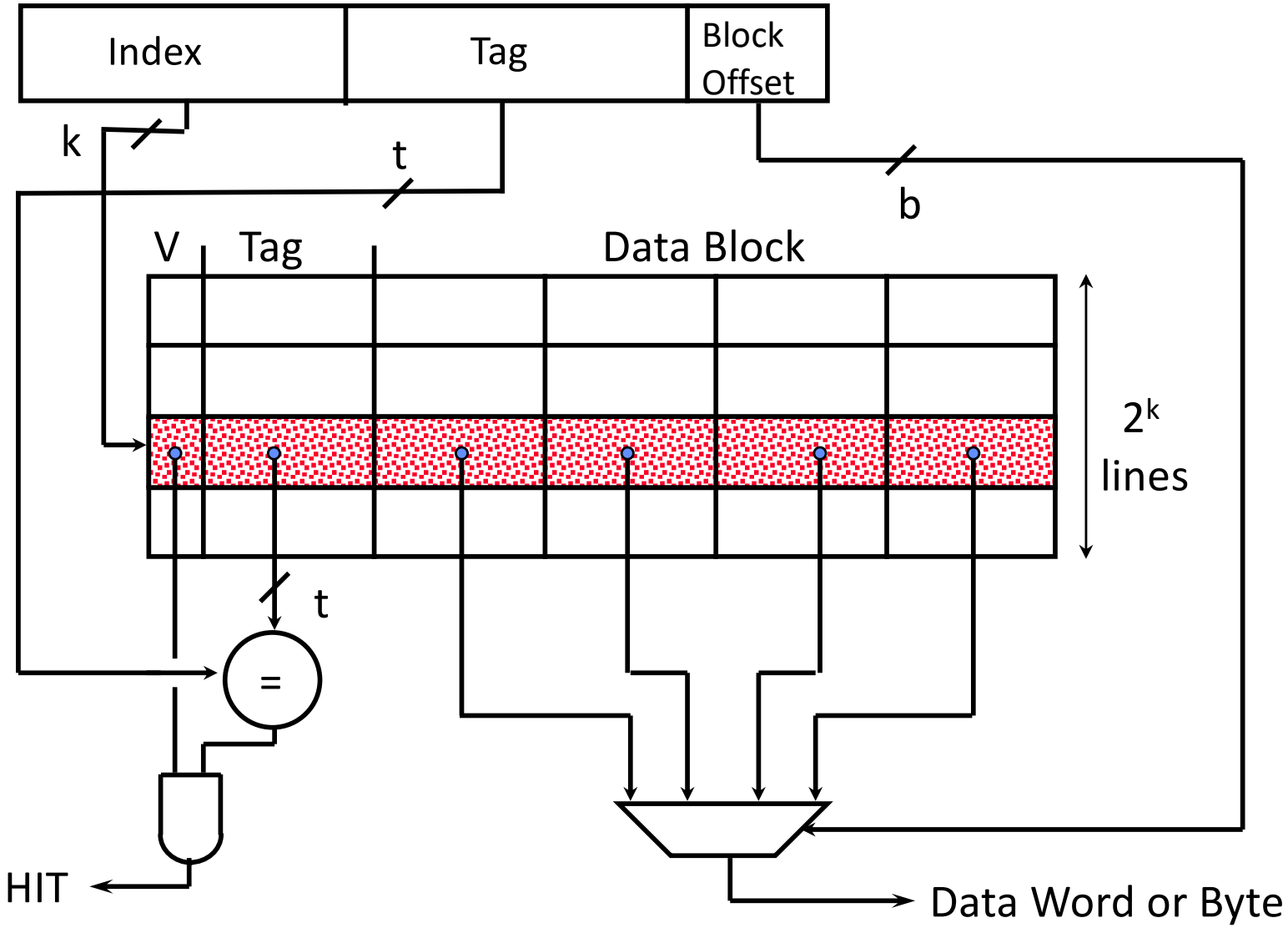


Direct-Mapped Cache

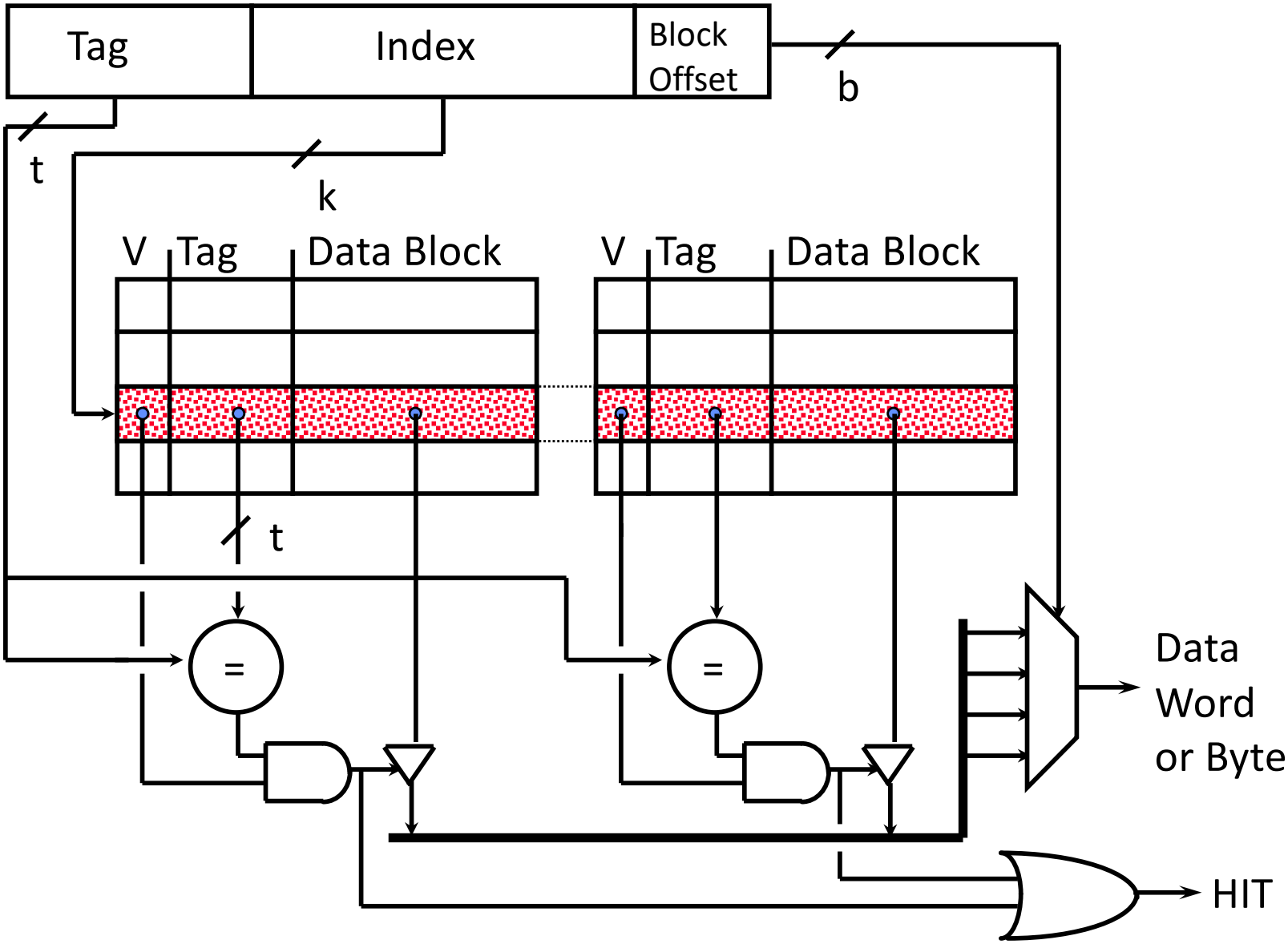


Direct Map Address Selection

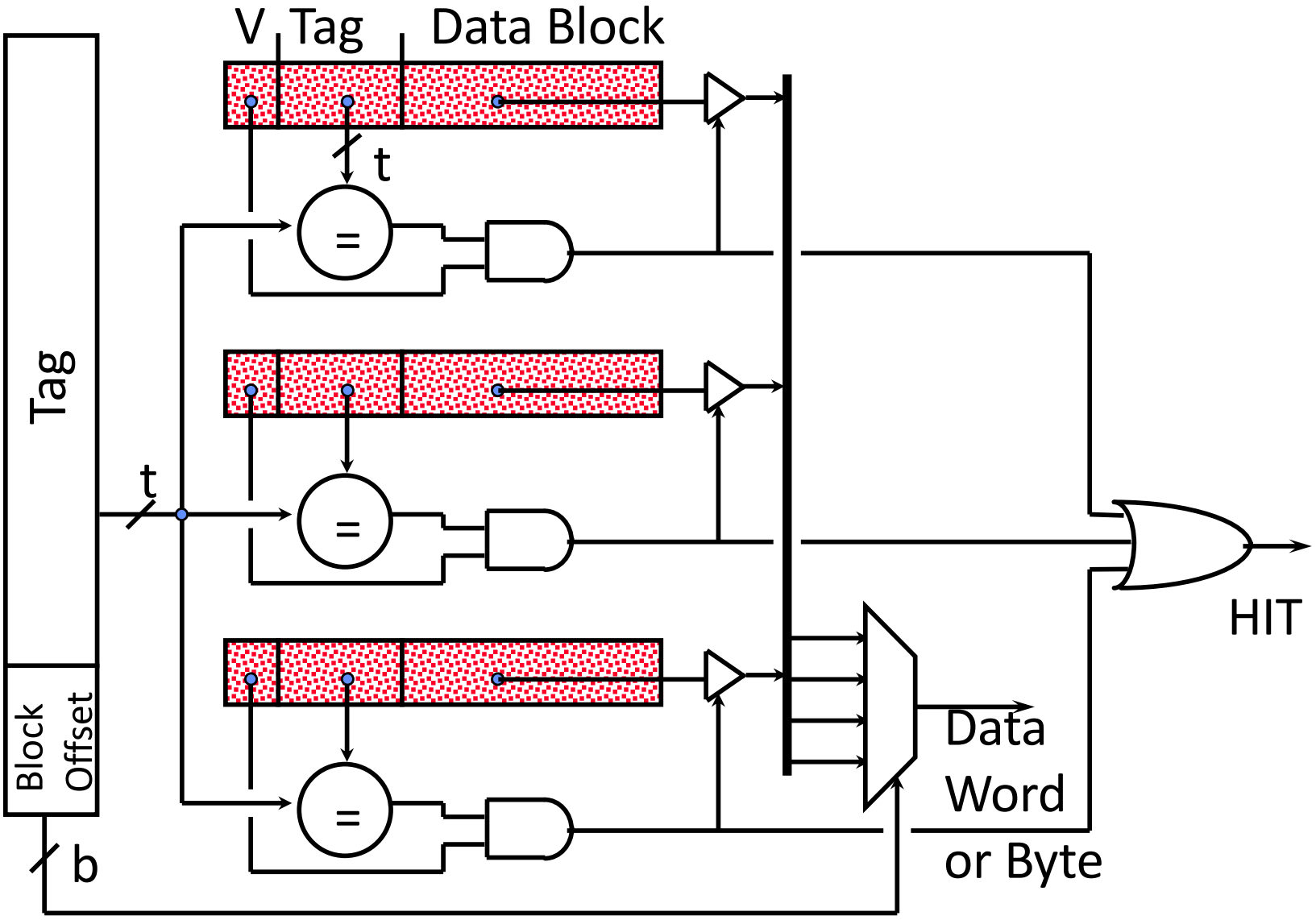
higher-order vs. lower-order address bits



2-Way Set-Associative Cache



Fully Associative Cache



Replacement Policy

In an associative cache, which block from a set should be evicted when the set becomes full?

- Random
- Least-Recently Used (LRU)
  - LRU cache state must be updated on every access
  - true implementation only feasible for small sets (2-way)
  - pseudo-LRU binary tree often used for 4-8 way
- First-In, First-Out (FIFO) a.k.a. Round-Robin
  - used in highly associative caches
- Not-Most-Recently Used (NMRU)
  - FIFO with exception for most-recently used block or blocks

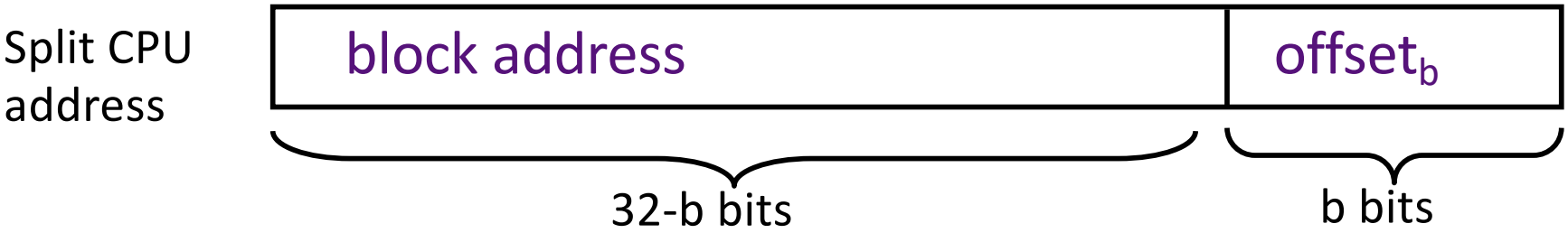
This is a second-order effect. Why?

Replacement only happens on misses

Block Size and Spatial Locality

Block is unit of transfer between the cache and memory

4 word block, b=2



2^b = block size *a.k.a* line size (in bytes)

Larger block size has distinct hardware advantages

- less tag overhead
- exploit fast burst transfers from DRAM
- exploit fast burst transfers over wide busses

What are the disadvantages of increasing block size?

Fewer blocks => more conflicts. Can waste bandwidth.

Acknowledgements

- This course is partly inspired by previous MIT 6.823 and Berkeley CS252 computer architecture courses created by my collaborators and colleagues:
  - Arvind (MIT)
  - Joel Emer (Intel/MIT)
  - James Hoe (CMU)
  - John Kubiatowicz (UCB)
  - David Patterson (UCB)