Lecture 10: Memory System
-- Memory Technology

CSE 564 Computer Architecture Summer 2017

Department of Computer Science and Engineering
Yonghong Yan
yan@oakland.edu
www.secs.oakland.edu/~yan

Topics for Memory Systems

- **Memory Technology and Metrics**
  - **SRAM, DRAM, Flash/SSD, 3-D Stack Memory, Phase-change memory**
  - **Latency and Bandwidth, Error Correction**
  - **Memory wall**
- Cache
  - Cache basics
  - Cache performance and optimization
  - Advanced optimization
  - Multiple-level cache, shared and private cache, prefetching
- Virtual Memory
  - Protection, Virtualization, and Relocation
  - Page/segment, protection
  - Address Translation and TLB

Topics for Memory Systems

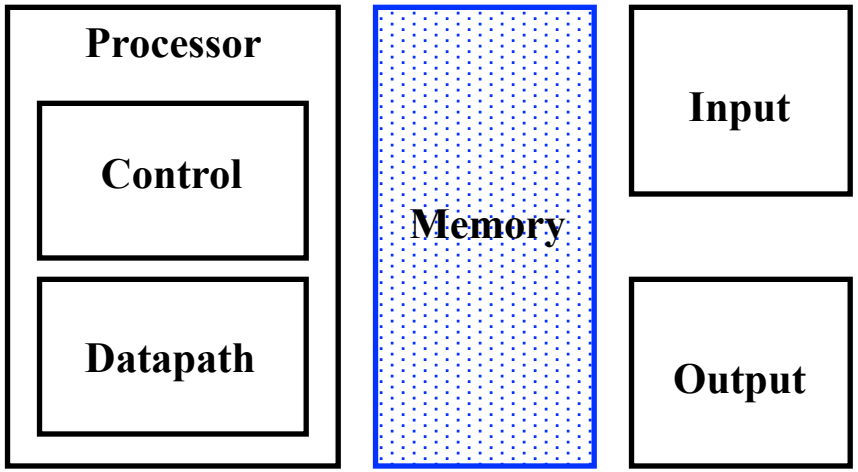
- Parallelism (to be discussed in TLP)
  - Memory Consistency model
    - Instructions for fence, etc
  - Cache coherence
  - NUMA and first touch
  - Transactional memory (Not covered)

- Implementation – (Not Covered)
  - Software/Hardware interface
  - Cache/memory controller
  - Bus systems and interconnect

Acknowledgement

- Based on slides prepared by: Professor David A. Patterson Computer Science 252, Fall 1996, and edited and presented by Prof. Kurt Keutzer for 2000 from UCB
- Some slides are adapted from the textbook slides for Computer Organization and Design, Fifth Edition: The Hardware/Software Interface

The Big Picture: Where are We Now?



- Memory system
  - Supplying data on time for computation
  - Term **memory** include circuits for storing data
    - Cache (SRAM)
    - Scratchpad (SRAM)
    - RAM (DRAM)
    - etc

Technology Trends

	Capacity	Speed (latency)
Logic:	2x in 3 years	2x in 3 years
DRAM:	4x in 3 years	2x in 10 years
Disk:	4x in 3 years	2x in 10 years

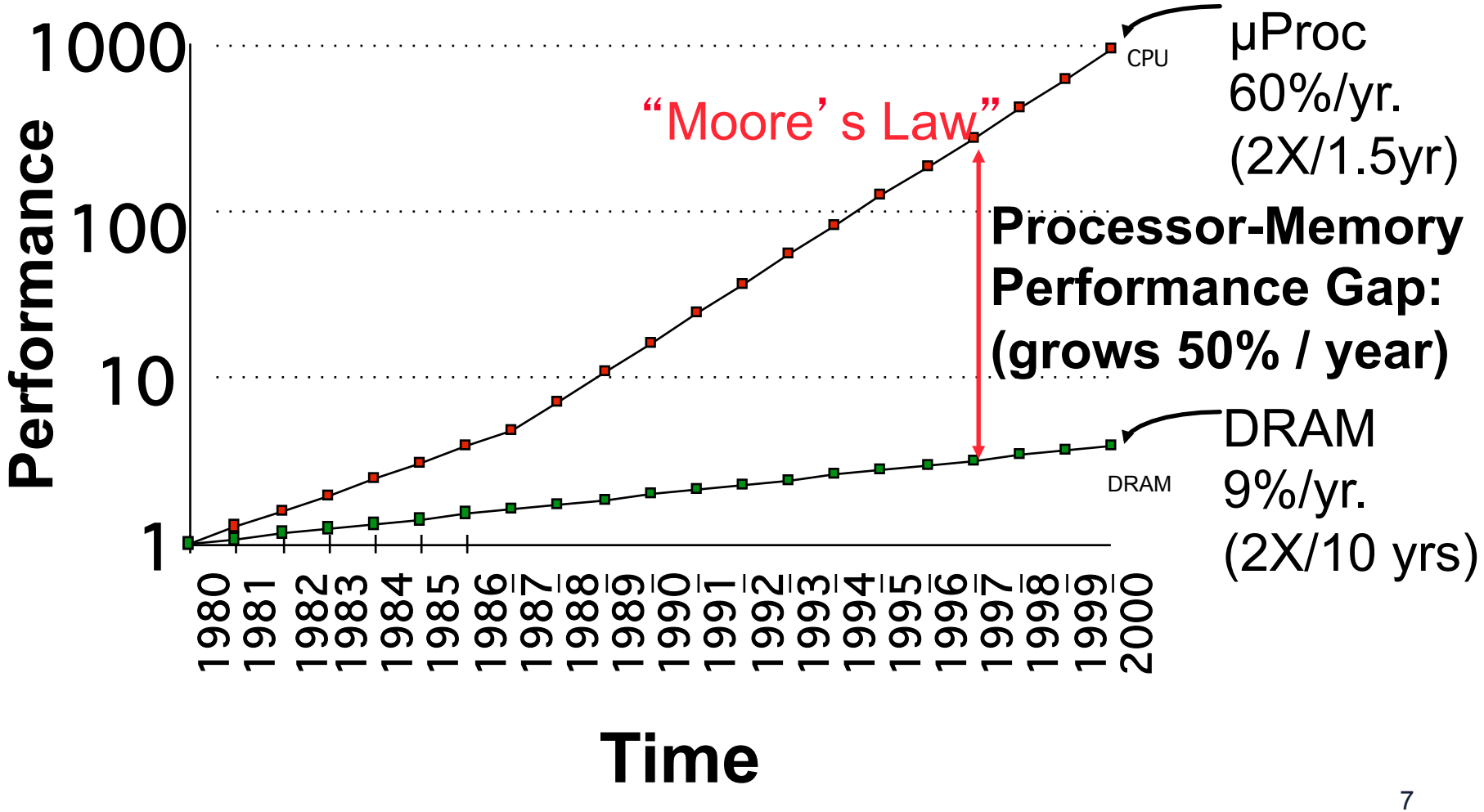
DRAM

Year	Size	Cycle Time
1980	64 Kb	250 ns
1983	256 Kb	220 ns
1986	1 Mb	190 ns
1989	4 Mb	165 ns
1992	16 Mb	145 ns
1995	64 Mb	120 ns

1000:1! 2:1!

Who Cares About the Memory Hierarchy?

Processor-DRAM Memory Gap (latency) → Memory Wall



The Situation: Microprocessor

- Rely on caches to bridge gap
- Microprocessor-DRAM performance gap
  - time of a full cache miss in instructions executed

1st Alpha (7000): 340 ns/5.0 ns = 68 clks x 2 or 136 instructions

2nd Alpha (8400): 266 ns/3.3 ns = 80 clks x 4 or 320 instructions

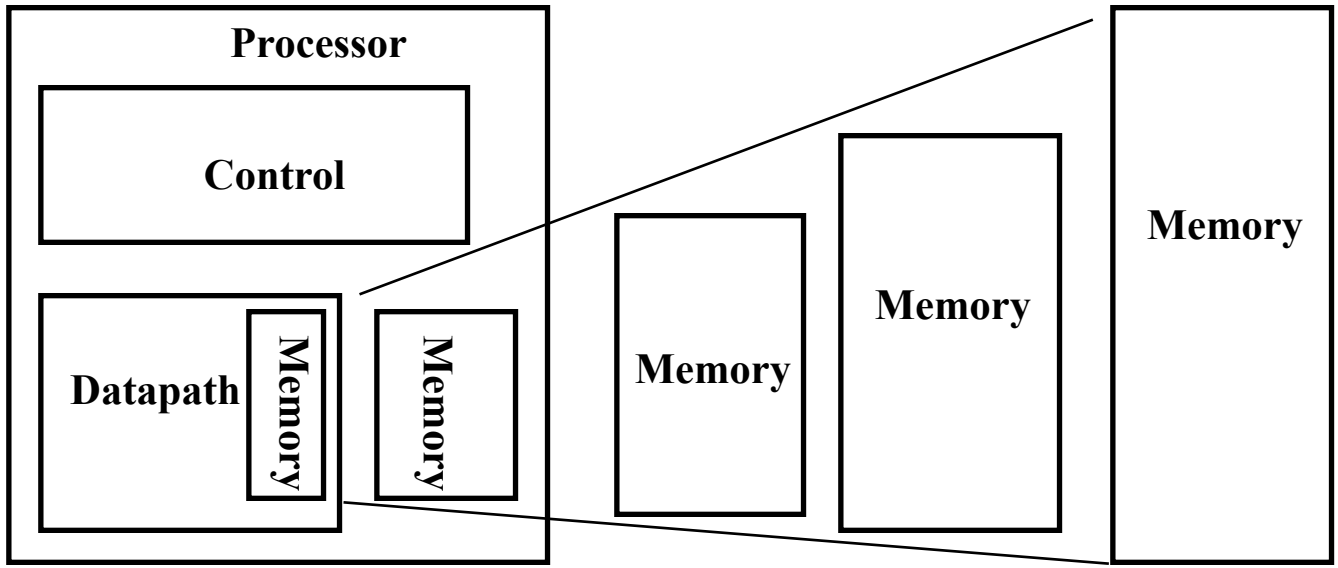
3rd Alpha (t.b.d.): 180 ns/1.7 ns =108 clks x 6 or 648 instructions

  - 1/2X latency x 3X clock rate x 3X Instr/clock $\Rightarrow$ -5X

The Goal: illusion of large, fast, cheap memory

- Fact: Large memories are slow, fast memories are small
- How do we create a memory that is large, cheap and fast (most of the time)?
  - Hierarchy
  - Parallelism

An Expanded View of the Memory System

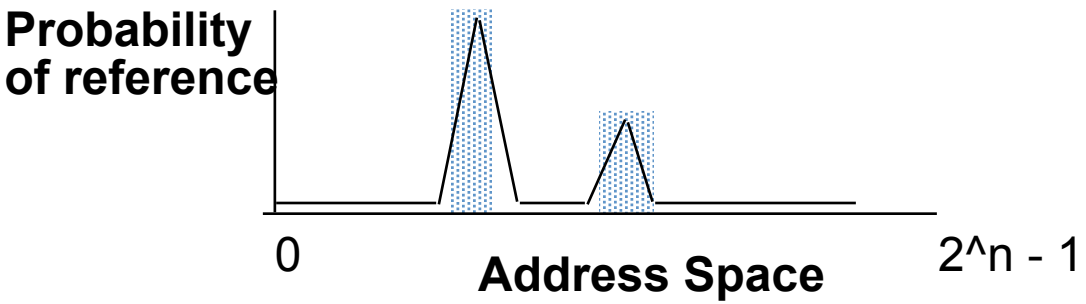


Speed: Fastest
Size: Smallest
Cost: Highest

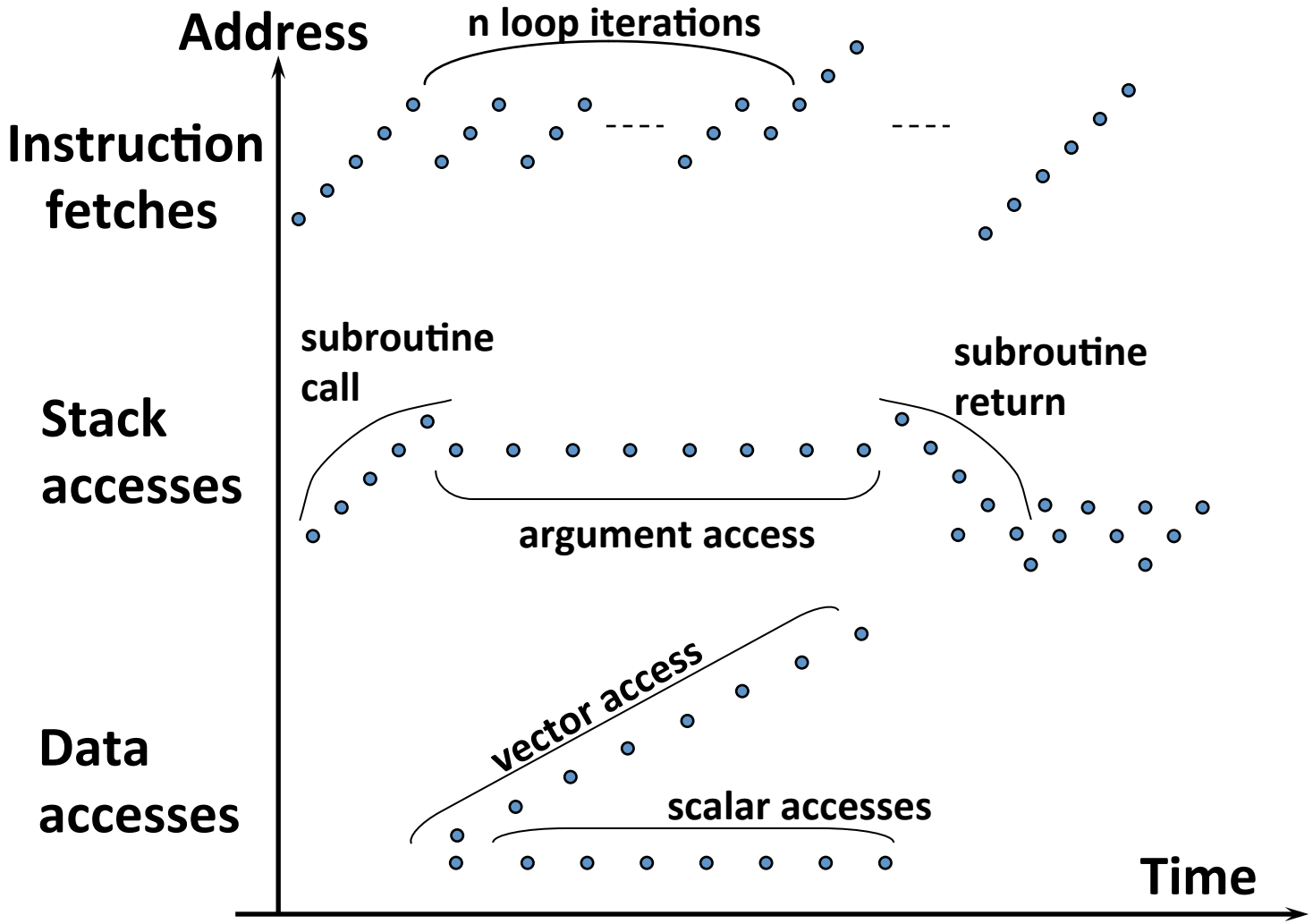
Slowest
Biggest
Lowest

Why hierarchy works

- The Principle of Locality:
  - Program access a relatively small portion of the address space at any instant of time.



Typical Memory Reference Patterns



Locality

- Principle of Locality:
  - Programs tend to reuse data and instructions near those they have used recently, or that were recently referenced themselves
  - Spatial locality: Items with nearby addresses tend to be referenced close together in time
  - Temporal locality: Recently referenced items are likely to be referenced in the near future

Locality Example:

```
sum = 0;
for (i = 0; i < n; i++)
    sum += a[i];
return sum;
```

- Data
  - Reference array elements in succession (stride-1 reference pattern): Spatial locality
  - Reference `sum` each iteration: Temporal locality
- Instructions
  - Reference instructions in sequence: Spatial locality
  - Cycle through loop repeatedly: Temporal locality

Locality Example

- Claim: Being able to look at code and get qualitative sense of its locality is key skill for professional programmer

- Question: Does this function have good locality?

```
int sumarrayrows(int a[M][N])
{
    int i, j, sum = 0;

    for (i = 0; i < M; i++)
        for (j = 0; j < N; j++)
            sum += a[i][j];
    return sum;
}
```

Locality Example

- Question: Does this function have good locality?

```
int sumarraycols(int a[M][N])
{
    int i, j, sum = 0;

    for (j = 0; j < N; j++)
        for (i = 0; i < M; i++)
            sum += a[i][j];
    return sum;
}
```

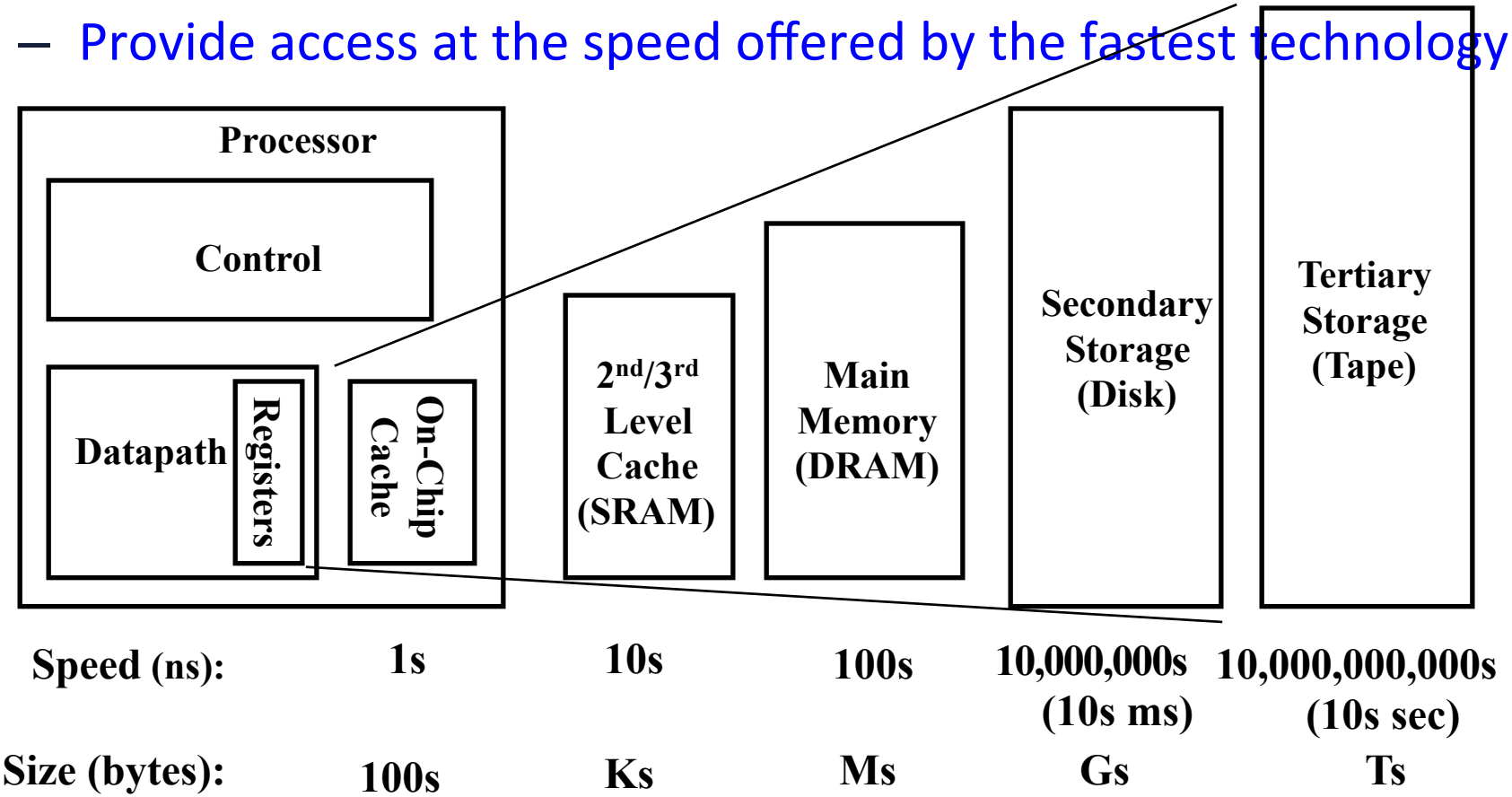
Locality Example

- Question: Can you permute the loops so that the function scans the 3-d array `a[]` with a stride-1 reference pattern (and thus has good spatial locality)?

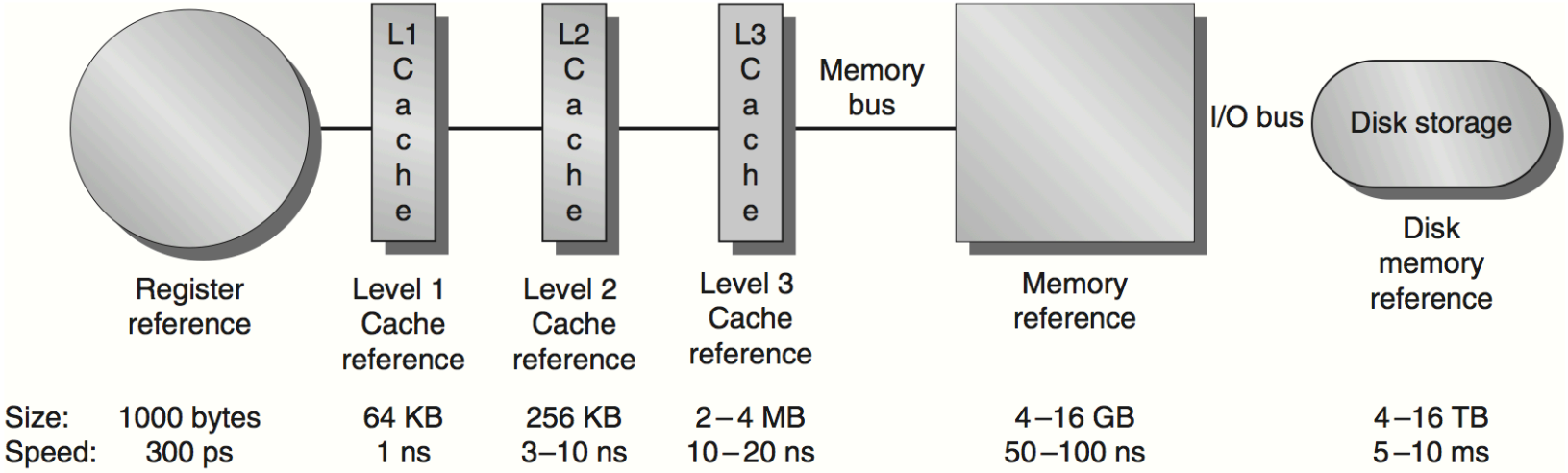
```
int sumarray3d(int a[M][N][N])
{
    int i, j, k, sum = 0;

    for (i = 0; i < N; i++)
        for (j = 0; j < N; j++)
            for (k = 0; k < M; k++)
                sum += a[k][i][j];
    return sum;
}
```

Memory Hierarchy of a Computer System

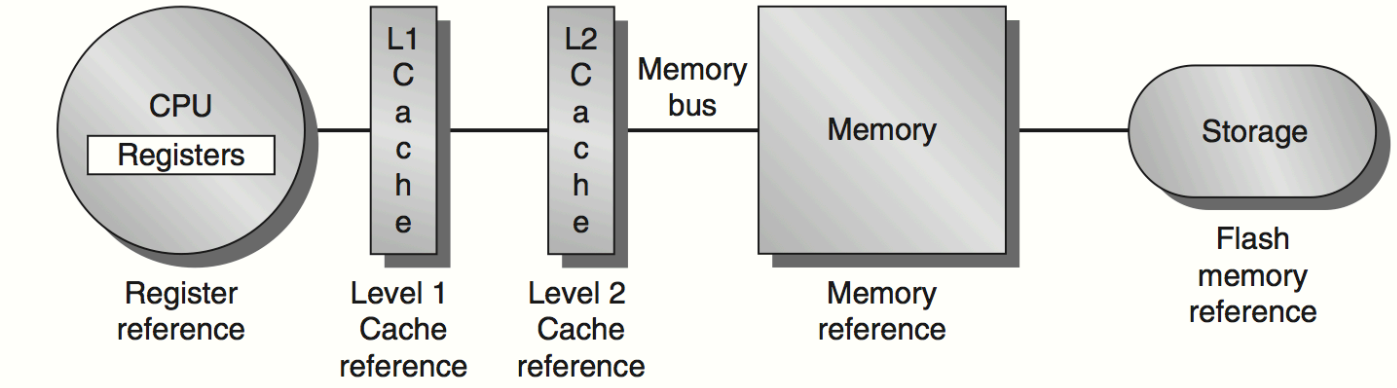
- By taking advantage of the principle of locality:
  - Present the user with as much memory as is available in the cheapest technology.
  - Provide access at the speed offered by the fastest technology.



Memory Hierarchy in Real



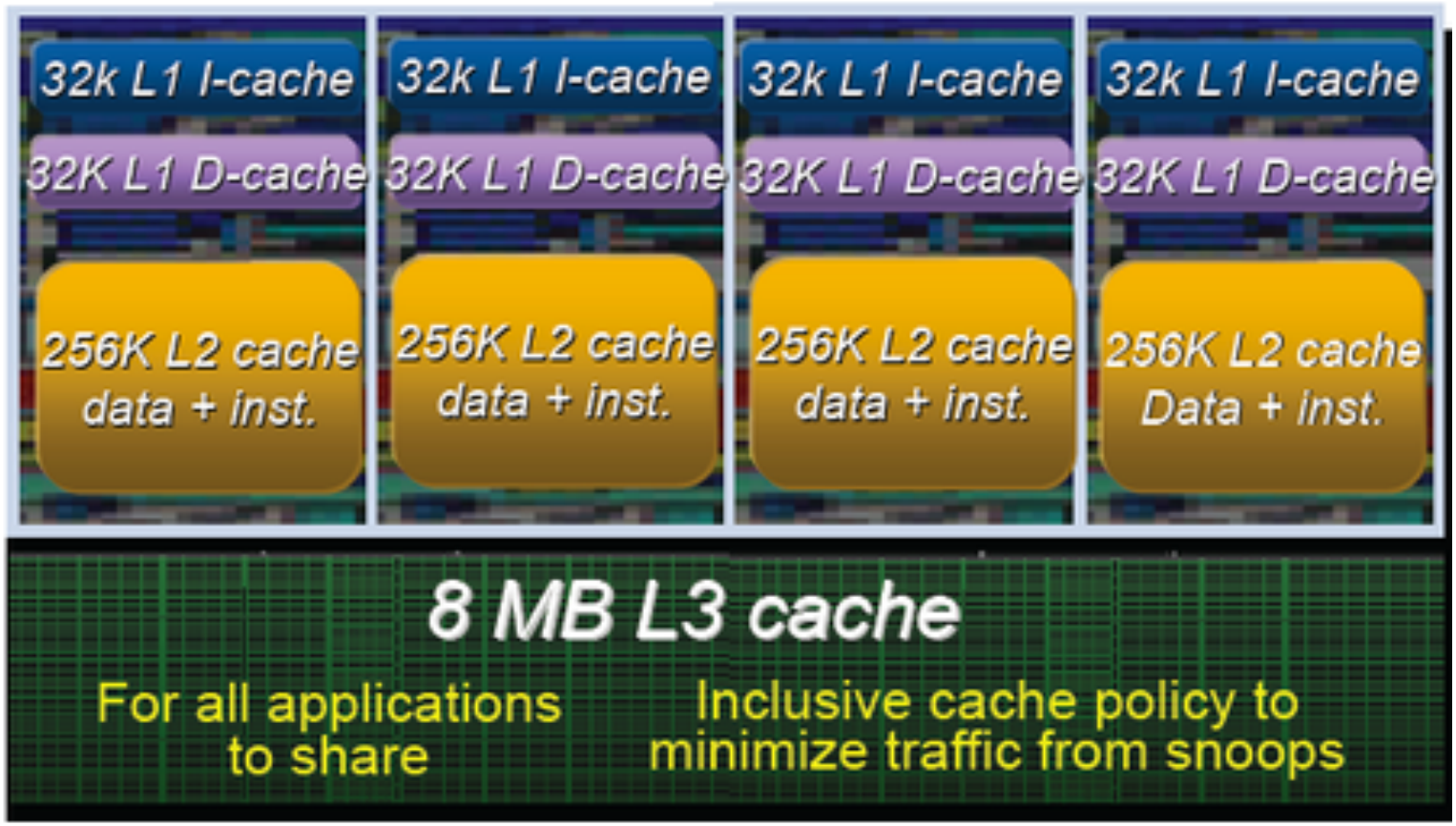
(a) Memory hierarchy for server



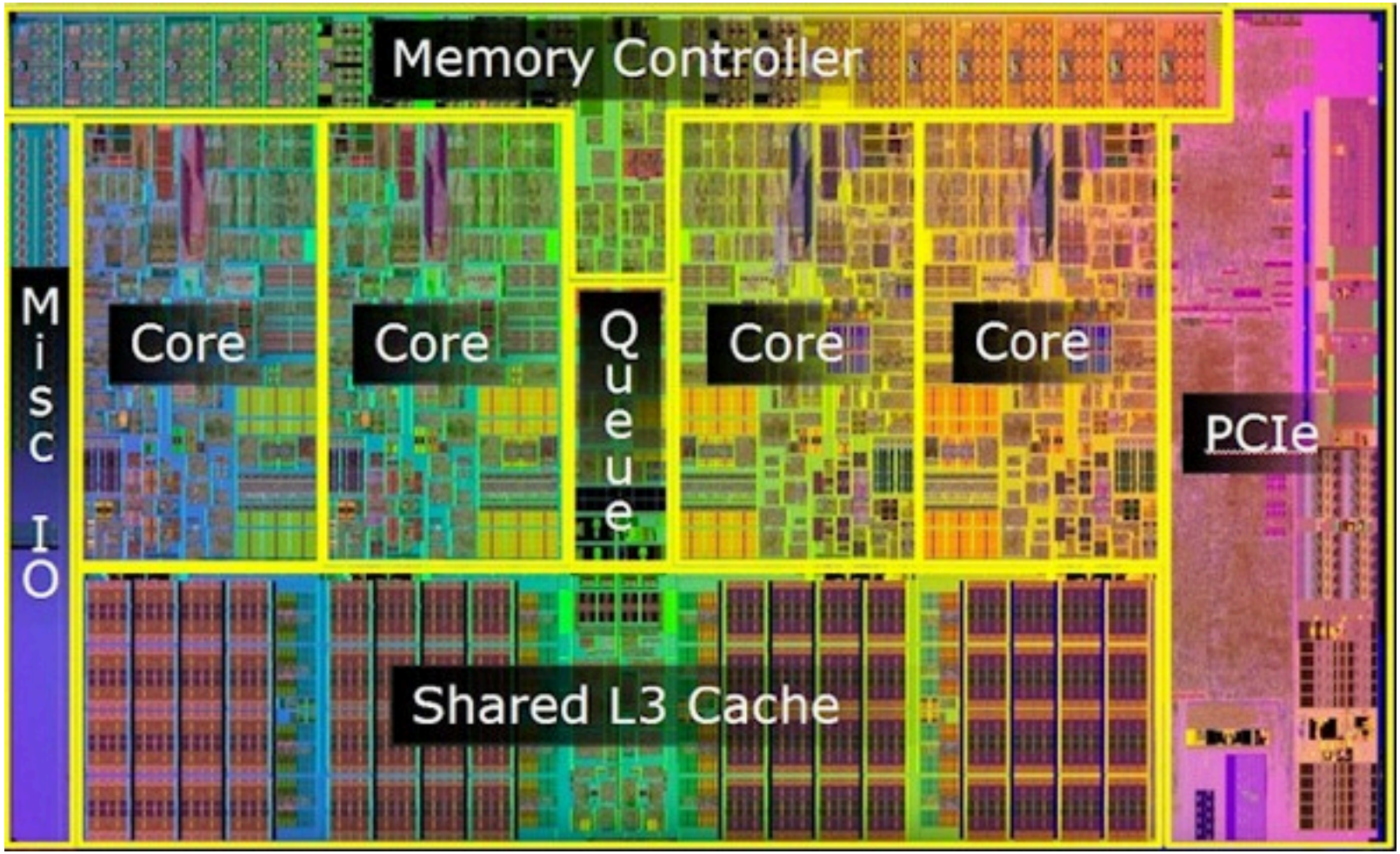
(b) Memory hierarchy for a personal mobile device

Intel i7 (Nahelem)

- Private L1 and L2
  - L2 is 256KB each. 10 cycle latency
- 8MB shared L3. ~40 cycles latency



Area


Memory Controller
Misc IO
Core
Core
Queue
Core
Core
PCIe
Shared L3 Cache

How is the hierarchy managed?

- Registers <-> Memory
  - by compiler (programmer?)
- cache <-> memory
  - by the hardware
- memory <-> disks
  - by the hardware and operating system (virtual memory)
  - by the programmer (files)

- Virtual memory
  - Virtual layer between application address space to physical memory
  - Not part of the physical memory hierarchy

Technology vs Architectures

- Technology determines the raw speed
  - Latency
  - Bandwidth
  - It is material science

- Architecture put them together with processors
  - So physical speed can be achieved in real

Memory Technology

- **Static RAM (SRAM)**
  - **0.5ns – 2.5ns, $2000 – $5000 per GB**
- **Dynamic RAM (DRAM)**
  - **50ns – 70ns, $20 – $75 per GB**
- 3-D stack memory
- Solid state disk
- Magnetic disk
  - 5ms – 20ms, $0.20 – $2 per GB

Ideal memory:
- **Access time of SRAM**
- **Capacity and cost/GB of disk**

Memory Technology

- Random Access:
  - "Random" is good: access time is the same for all locations
  - **DRAM: Dynamic Random Access Memory**
    - High density, low power, cheap, slow
    - Dynamic: need to be "refreshed" regularly
  - **SRAM: Static Random Access Memory**
    - Low density, high power, expensive, fast
    - Static: content will last "forever"(until lose power)
- "Non-so-random" Access Technology:
  - Access time varies from location to location and from time to time
  - Examples: Disk, CDROM
- Sequential Access Technology: access time linear in location (e.g.,Tape)

Main Memory Background

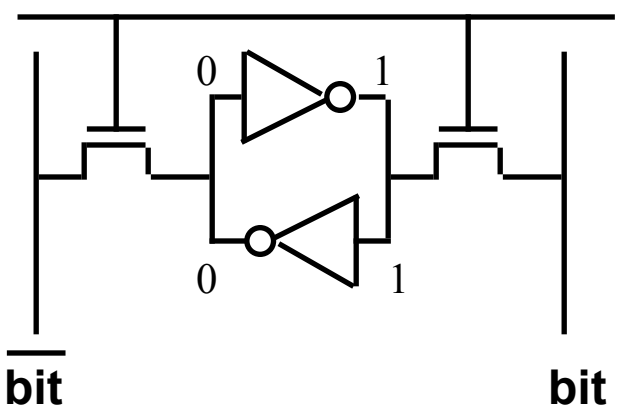
- Performance of Main Memory:
  - Latency: Cache Miss Penalty
    - *Access Time*: time between request and word arrives
    - *Cycle Time*: time between requests
  - Bandwidth: I/O & Large Block Miss Penalty (L2)
- Main Memory is *DRAM :* **Dynamic** Random Access Memory
  - Needs to be refreshed periodically (8 ms)
  - Addresses divided into 2 halves (Memory as a 2D matrix):
    - *RAS* or *Row Access Strobe*
    - *CAS* or *Column Access Strobe*
- Cache uses *SRAM :* **Static** Random Access Memory
  - No refresh (6 transistors/bit vs. 1 transistor)
    Size: DRAM/SRAM - *4-8*
    Cost/Cycle time: SRAM/DRAM - *8-16*

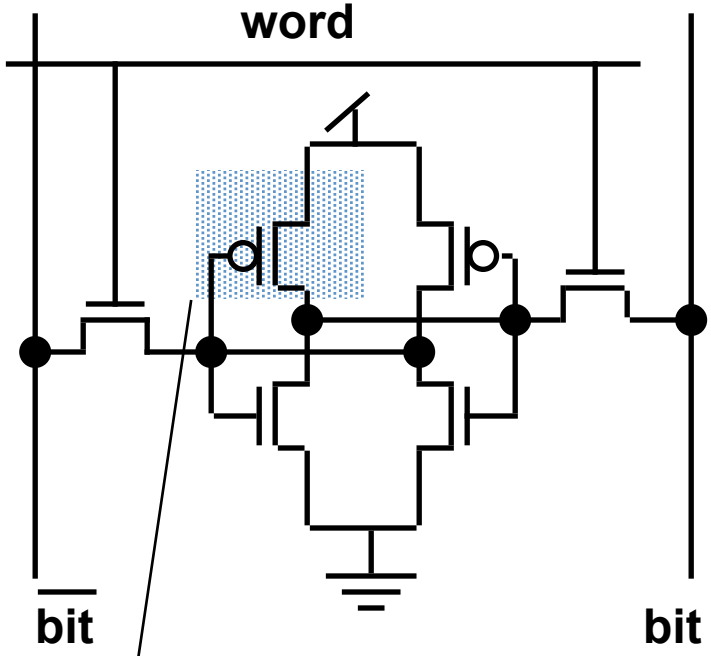
Random Access Memory (RAM) Technology

- Why need to know about RAM technology?
  - Processor performance is usually limited by memory bandwidth
  - As IC densities increase, lots of memory will fit on processor chip
    - Tailor on-chip memory to specific needs
      - **Instruction cache**
      - **Data cache**
      - **Write buffer**
- What makes RAM different from a bunch of **flip-flops**?
  - Density: RAM is much denser

Static RAM Cell

6-Transistor SRAM Cell



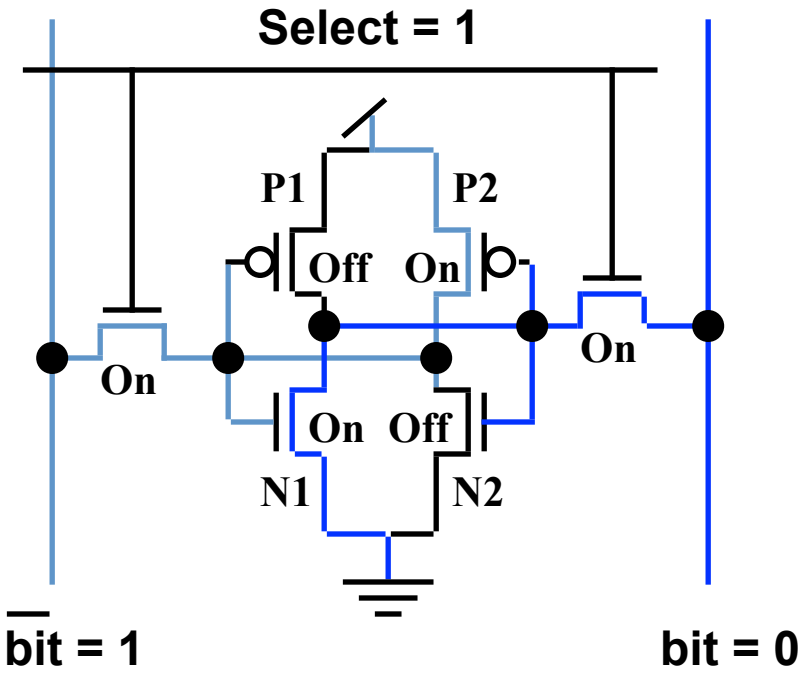


replaced with pullup to save area

- Write:
  1. Drive bit lines (bit=1, bit=0)
  2.. Select row
- Read:
  1. Precharge bit and $\overline{bit}$ to Vdd or Vdd/2 => make sure equal!
  2.. Select row
  3. Cell pulls one line low
  4. Sense amp on column detects difference between bit and $\overline{bit}$

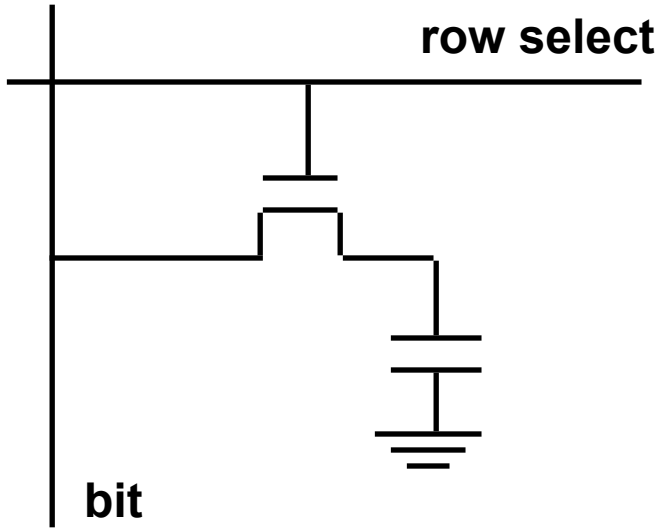
Problems with SRAM

- Six transistors use up a lot of area
- Consider a "Zero" is stored in the cell:
  - Transistor N1 will try to pull "bit" to 0
  - Transistor P2 will try to pull "bit bar" to 1
- But bit lines are precharged to high: Are P1 and P2 necessary?

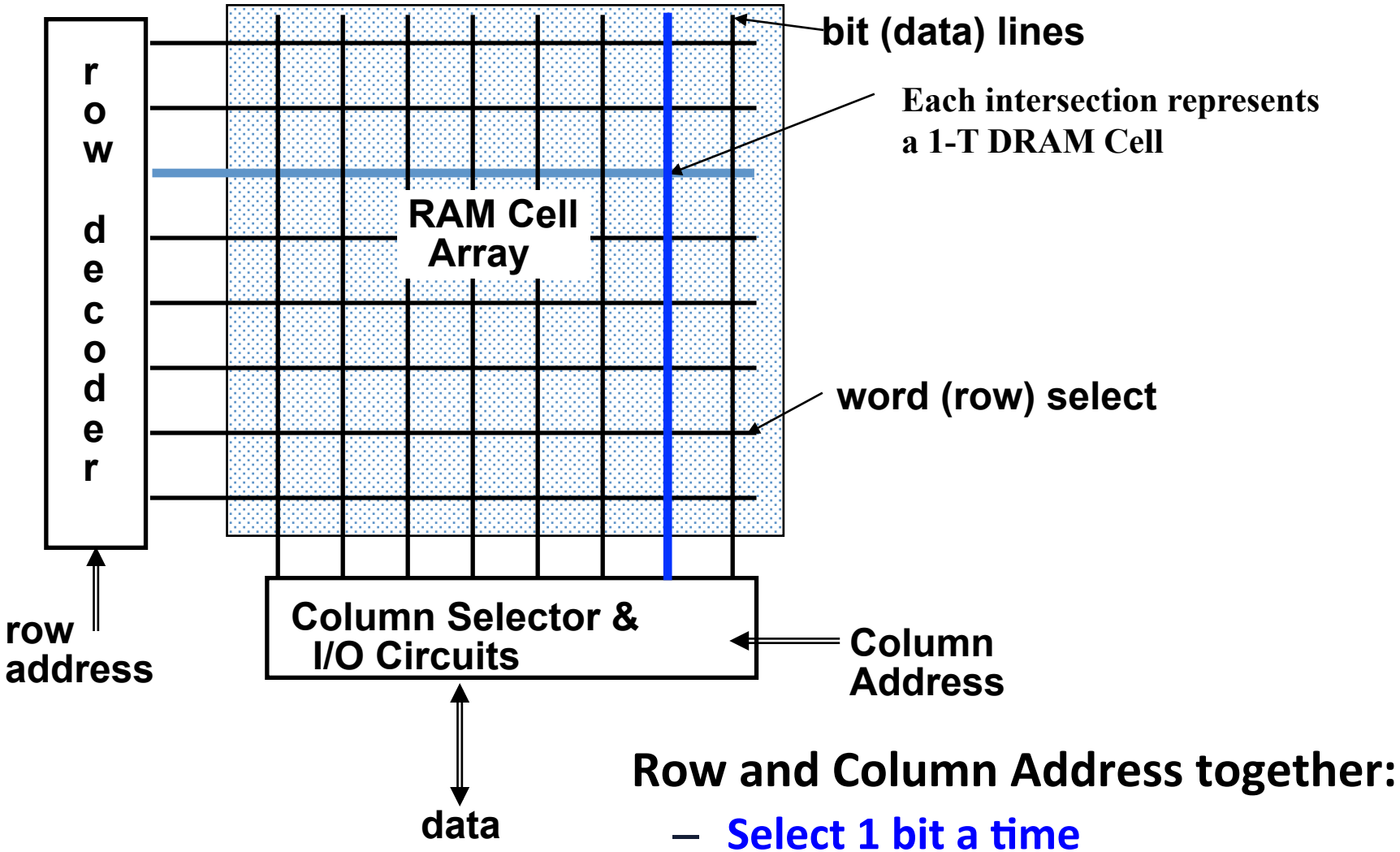


1-Transistor Memory Cell (DRAM)

- Write:
  - 1. Drive bit line
  - 2.. Select row
- Read:
  - 1. Precharge bit line to Vdd
  - 2.. Select row
  - 3. Cell and bit line share charges
    - Very small voltage changes on the bit line
  - 4. Sense (fancy sense amp)
    - Can detect changes of ~1 million electrons
  - 5. Write: restore the value
- Refresh
  - 1. Just do a dummy read to every cell.



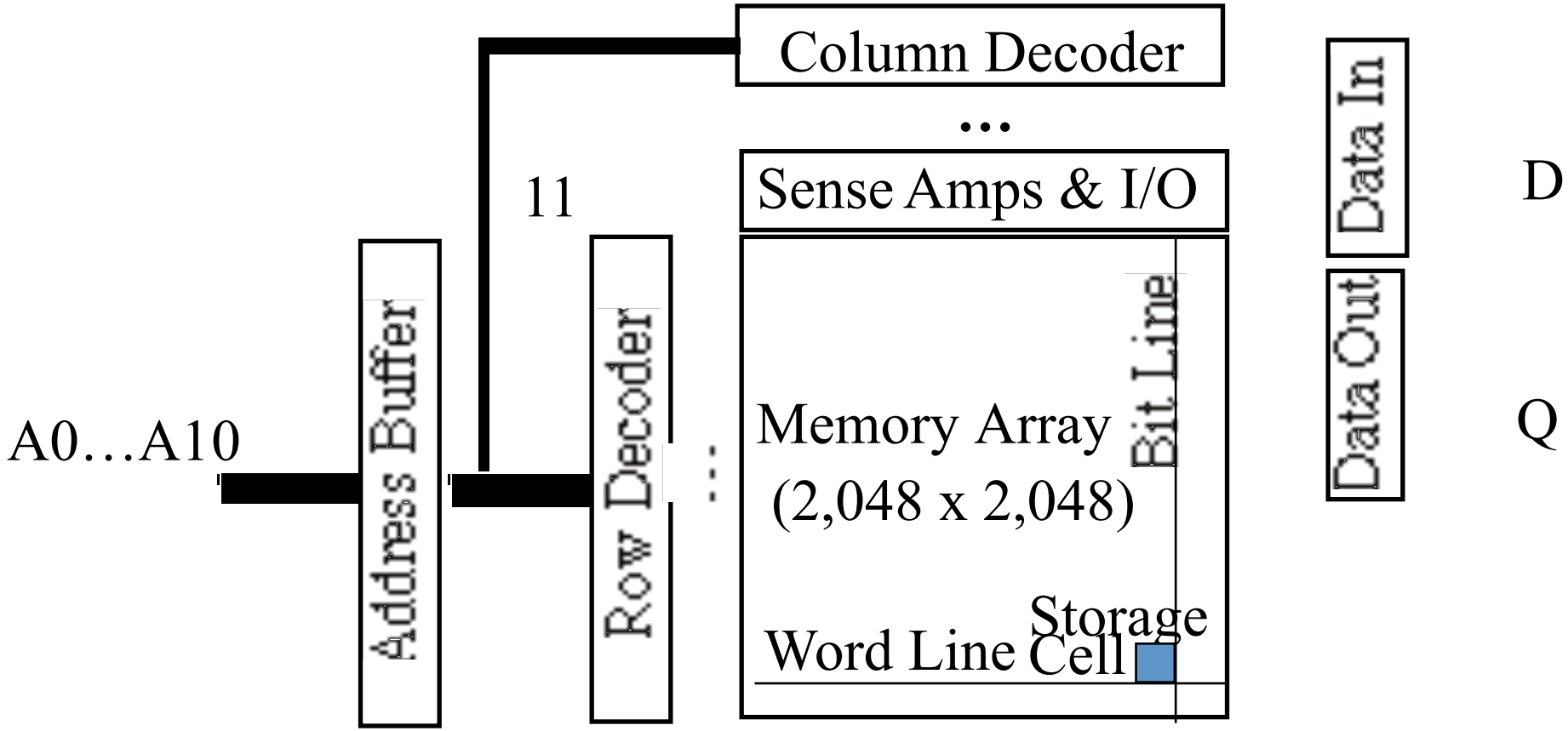
Classical DRAM Organization (square)



Row and Column Address together:

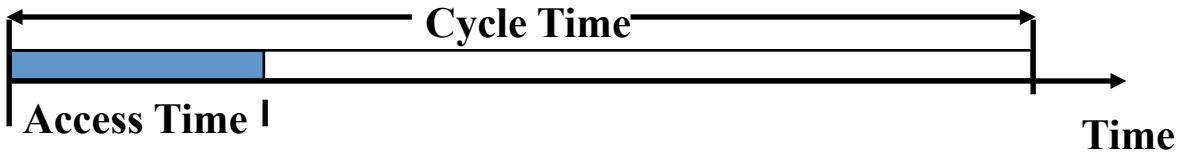
- **Select 1 bit a time**

DRAM logical organization (4 Mbit)



Square root of bits per RAS/CAS

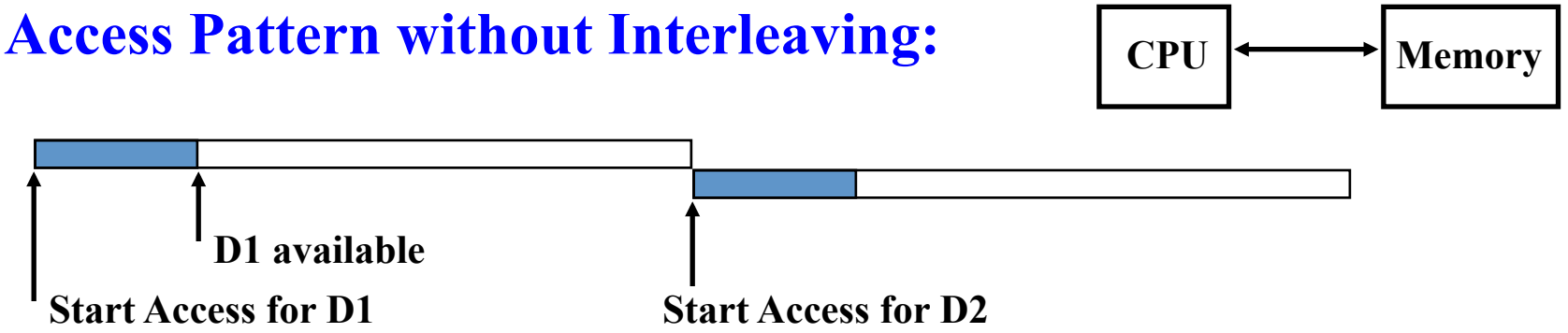
Main Memory Performance



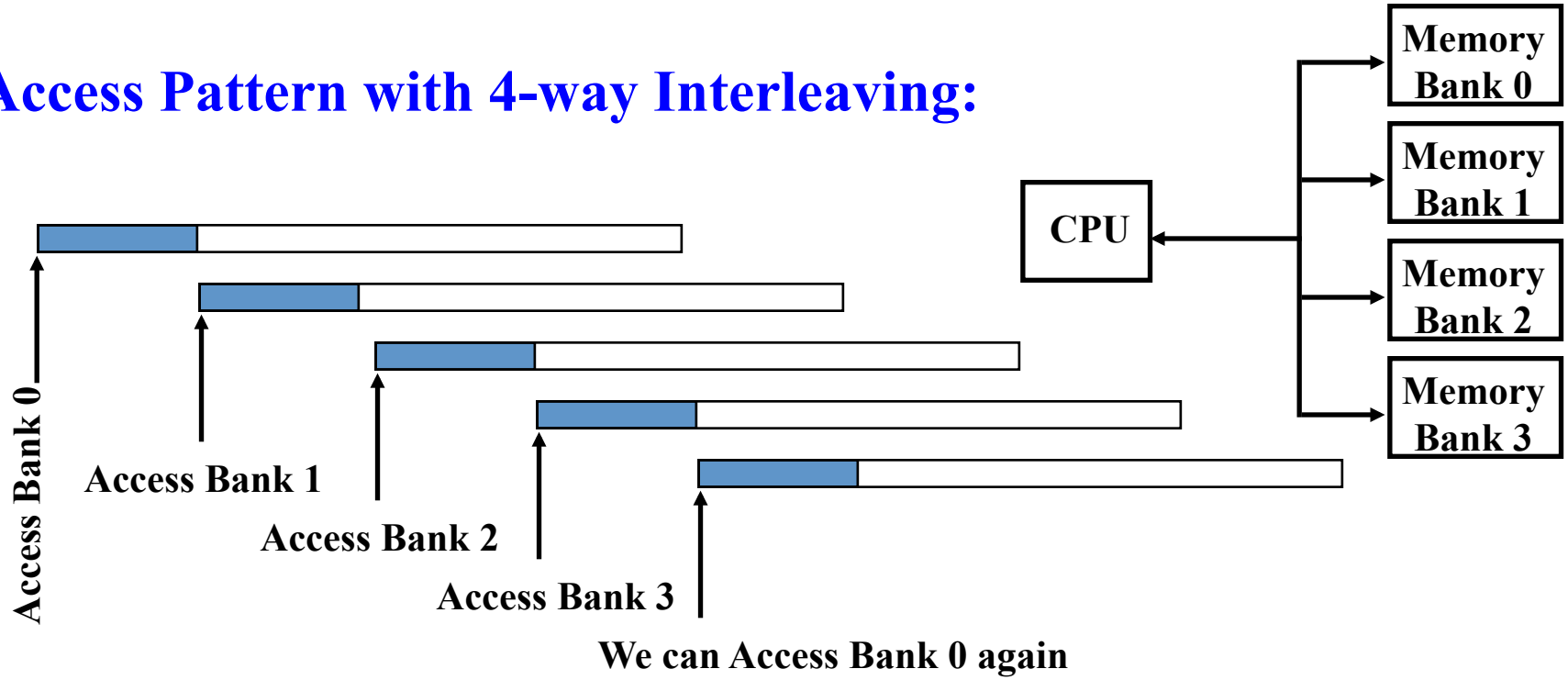
- DRAM (Read/Write) Cycle Time >> DRAM (Read/Write) Access Time
  - - 2:1
- DRAM (Read/Write) Cycle Time :
  - How frequent can you initiate an access?
  - Analogy: A little kid can only ask his father for money on Saturday
- DRAM (Read/Write) Access Time:
  - How quickly will you get what you want once you initiate an access?
  - Analogy: As soon as he asks, his father will give him the money
- DRAM Bandwidth Limitation analogy:
  - What happens if he runs out of money on Wednesday?

Increasing Bandwidth - Interleaving

Access Pattern without Interleaving:

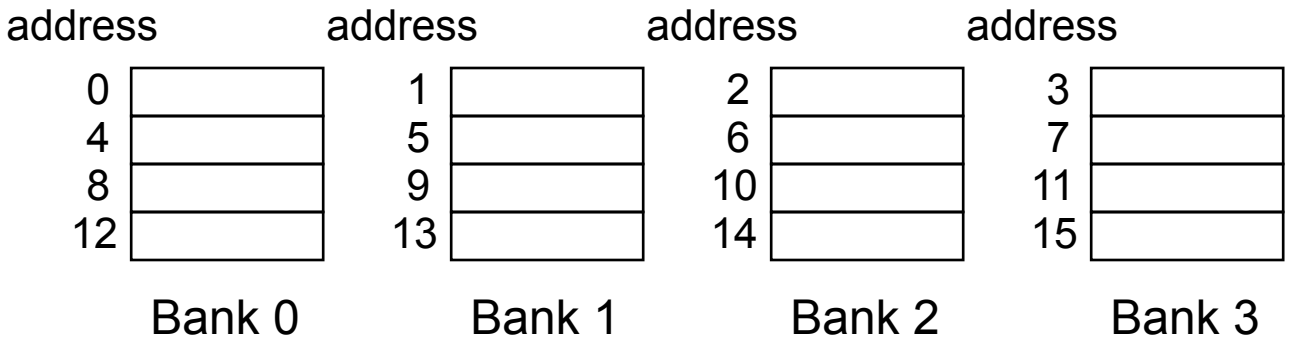


Access Pattern with 4-way Interleaving:



Main Memory Performance

- Timing model
  - 1 to send address,
  - 4 for access time, 10 cycle time, 1 to send data
  - Cache Block is 4 words
- *Simple M.P.* = 4 x (1+10+1) = 48
- *Wide M.P.* = 1 + 10 + 1 = 12
- *Interleaved M.P.* = 1+10+1 + 3 =15



Independent Memory Banks

- How many banks?

  number banks number clocks to access word in bank

  – For sequential accesses, otherwise will return to original bank before it has next word ready

- Increasing DRAM => fewer chips => harder to have banks

  – Growth bits/chip DRAM : 50%-60%/yr

  – Nathan Myrvold M/S: mature software growth (33%/yr for NT) - growth MB/$ of DRAM (25%-30%/yr)

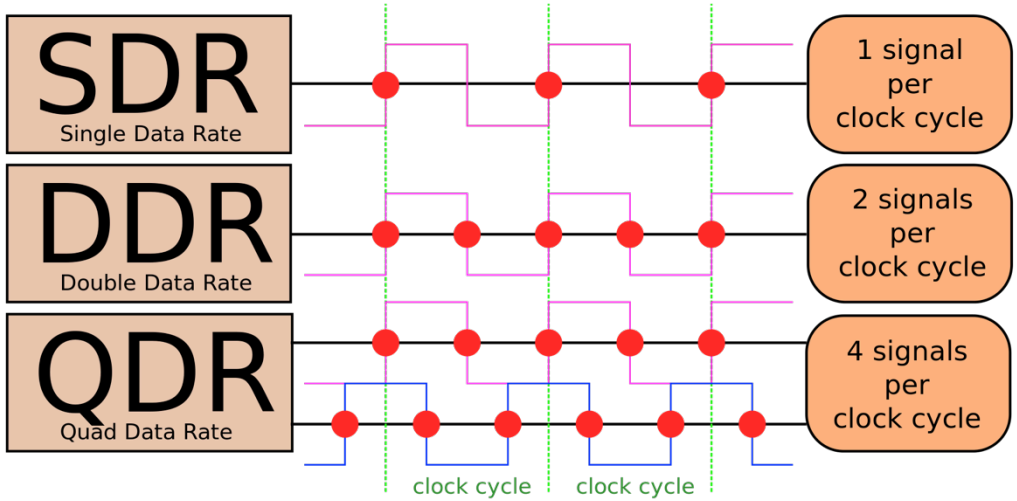
DRAM History

- DRAMs: capacity +60%/yr, cost –30%/yr
  - 2.5X cells/area, 1.5X die size in -3 years
- ‘97 DRAM fab line costs $1B to $2B
  - DRAM only: density, leakage v. speed
- Rely on increasing no. of computers & memory per computer (60% market)
  - SIMM or DIMM is replaceable unit
    => computers use any generation DRAM
- Commodity, second source industry
  => high volume, low profit, conservative
  - Little organization innovation in 20 years
    page mode, EDO, Synch DRAM
- Order of importance: 1) Cost/bit 1a) Capacity
  - RAMBUS: 10X BW, +30% cost => little impact

Advanced DRAM Organization

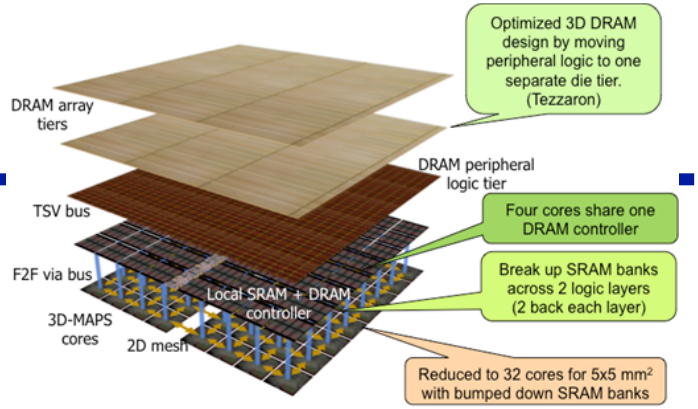
- Bits in a DRAM are organized as a rectangular array
  - DRAM accesses an entire row
  - Burst mode: supply successive words from a row with reduced latency

- Double data rate (DDR) DRAM
  - Transfer on rising and falling clock edges
- Quad data rate (QDR) DRAM
  - Separate DDR inputs and outputs



3-D Stack Memory

- **High Bandwidth Memory**
- Hybrid Memory Cube



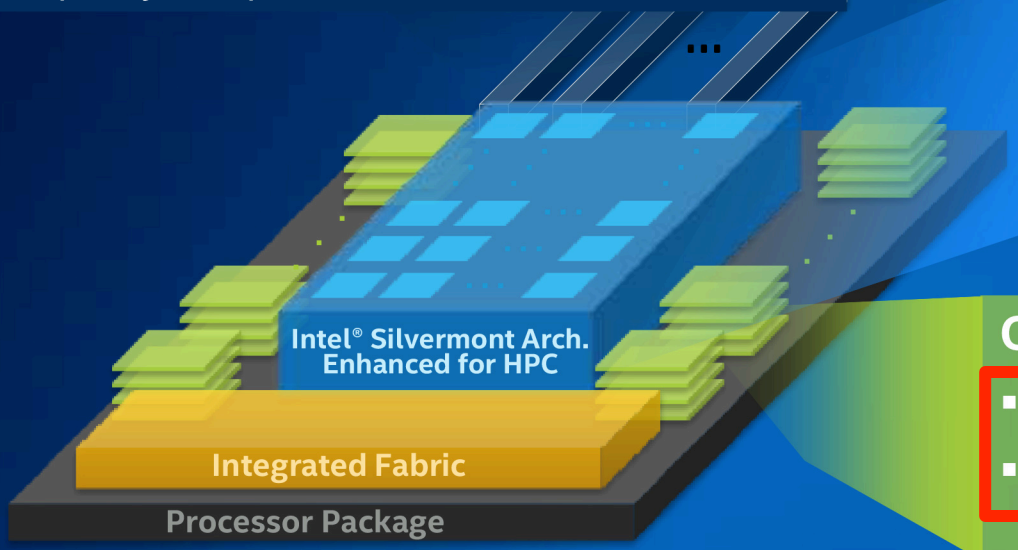
Unveiling Details of Knights Landing

(Next Generation Intel® Xeon Phi™ Products)

2nd half '15
1st commercial systems

3+ TFLOPS[1]
In One Package
Parallel Performance & Density

Platform Memory: DDR4 Bandwidth and Capacity Comparable to Intel® Xeon® Processors



Compute: Energy-efficient IA cores[2]

- *Microarchitecture enhanced for HPC[3]*
- ***3X*** *Single Thread Performance vs Knights Corner[4]*
- *Intel Xeon Processor Binary Compatible[5]*

On-Package Memory:

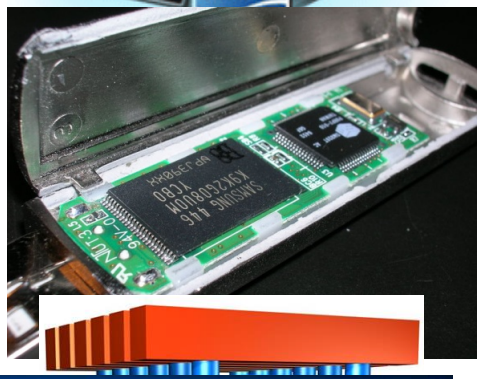
- *up to* ***16GB*** *at launch*
- ***1/3X*** *the Space[6]*
- ***5X*** *Bandwidth vs DDR4[7]*
- ***5X*** *Power Efficiency[6]*

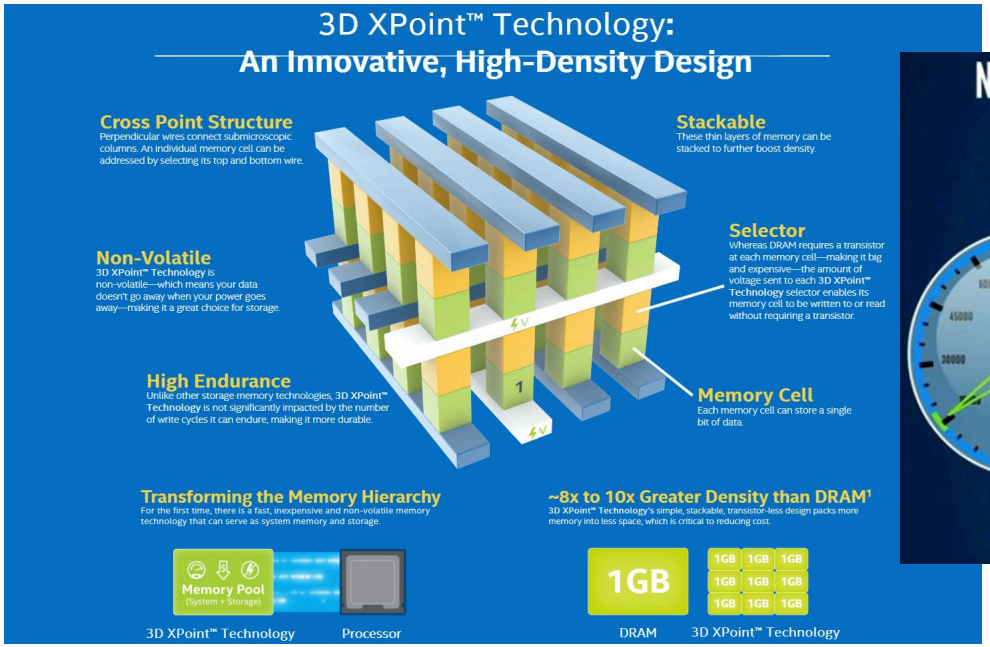
Jointly Developed with Micron Technology

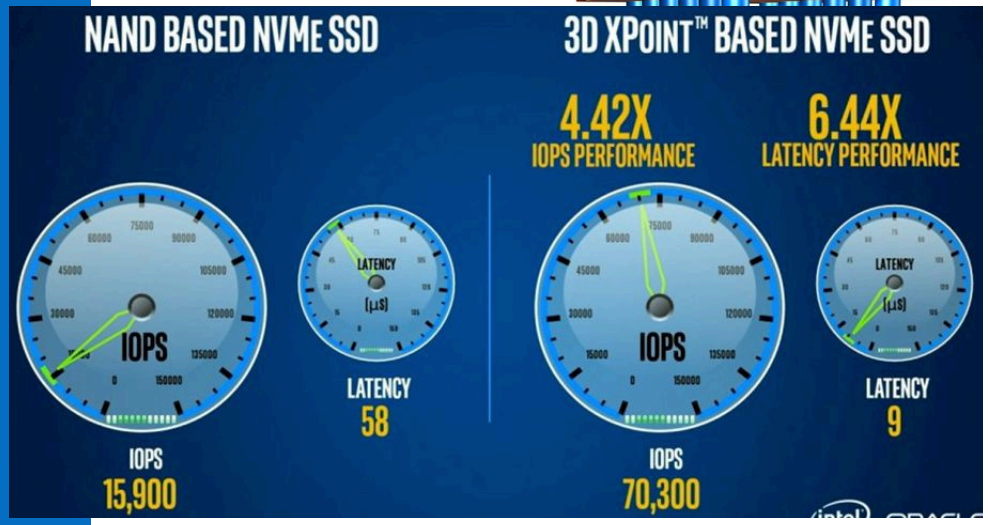
All products, computer systems, dates and figures specified are preliminary based on current expectations, and are subject to change without notice. [1]Over 3 Teraflops of peak theoretical double-precision performance is preliminary and based on current expectations of cores, clock frequency and floating point operations per cycle. FLOPS = cores x clock frequency x floating-

Flash Storage

- Nonvolatile semiconductor storage
  - 100× – 1000× faster than disk
  - Smaller, lower power, more robust
  - But more $/GB (between disk and DRAM)





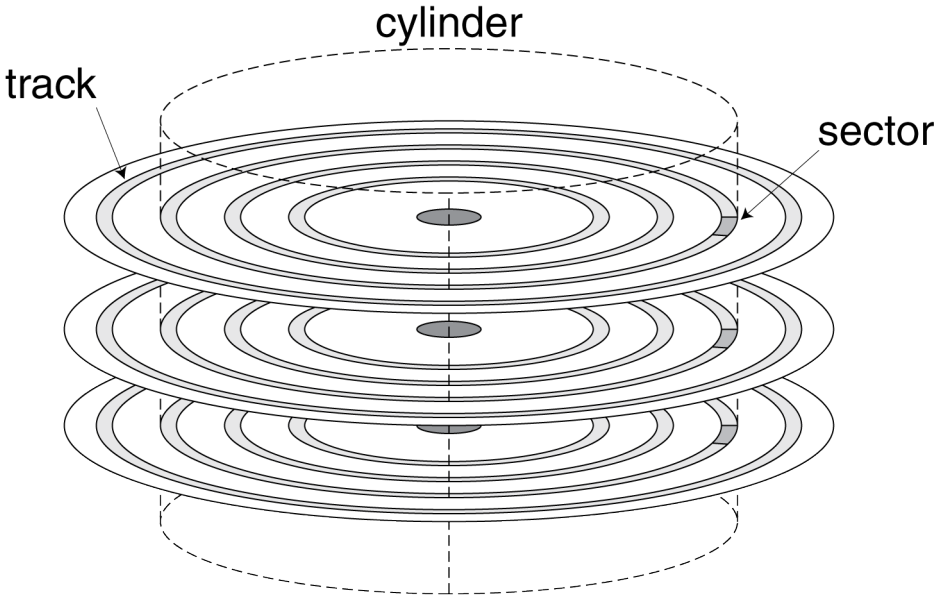


Flash Types

- NOR flash: bit cell like a NOR gate
  - Random read/write access
  - Used for instruction memory in embedded systems
- NAND flash: bit cell like a NAND gate
  - Denser (bits/area), but block-at-a-time access
  - Cheaper per GB
  - Used for USB keys, media storage, …
- Flash bits wears out after 1000's of accesses
  - Not suitable for direct RAM or disk replacement
  - Wear leveling: remap data to less used blocks

Disk Storage

- Nonvolatile, rotating magnetic storage



Disk Sectors and Access

- Each sector records
  - Sector ID
  - Data (512 bytes, 4096 bytes proposed)
  - Error correcting code (ECC)
    - Used to hide defects and recording errors
  - Synchronization fields and gaps
- Access to a sector involves
  - Queuing delay if other accesses are pending
  - Seek: move the heads
  - Rotational latency
  - Data transfer
  - Controller overhead

Disk Access Example

- Given
  - 512B sector, 15,000rpm, 4ms average seek time, 100MB/s transfer rate, 0.2ms controller overhead, idle disk
- Average read time
  - 4ms seek time
    + ½ / (15,000/60) = 2ms rotational latency
    + 512 / 100MB/s = 0.005ms transfer time
    + 0.2ms controller delay
    = 6.2ms
- If actual average seek time is 1ms
  - Average read time = 3.2ms

Summary:

- Two Different Types of Locality:
  - Temporal Locality (Locality in Time): If an item is referenced, it will tend to be referenced again soon.
  - Spatial Locality (Locality in Space): If an item is referenced, items whose addresses are close by tend to be referenced soon.
- By taking advantage of the principle of locality:
  - Present the user with as much memory as is available in the cheapest technology.
  - Provide access at the speed offered by the fastest technology.
- DRAM is slow but cheap and dense:
  - Good choice for presenting the user with a BIG memory system
- SRAM is fast but expensive and not very dense:
  - Good choice for providing the user FAST access time.