CPT_S 260 Intro to Computer Architecture
Lecture 39

Memory
April 20, 2022

Ganapati Bhat
School of Electrical Engineering and Computer Science
Washington State University

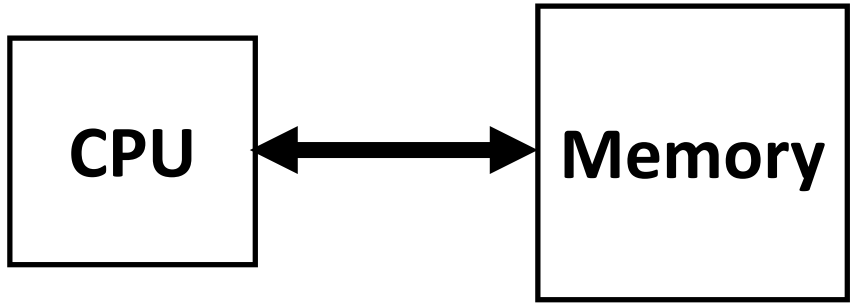
Announcements

- **Final exam**
  - May 4th 2022
  - Comprehensive with all topics
  - Review on Wednesday and Friday
- **Class evaluations are open**
  - Please complete the class review
  - Feedback will help in improving the course in the future
  - Included as part of class participation

Eight Great Ideas in Computer Architecture

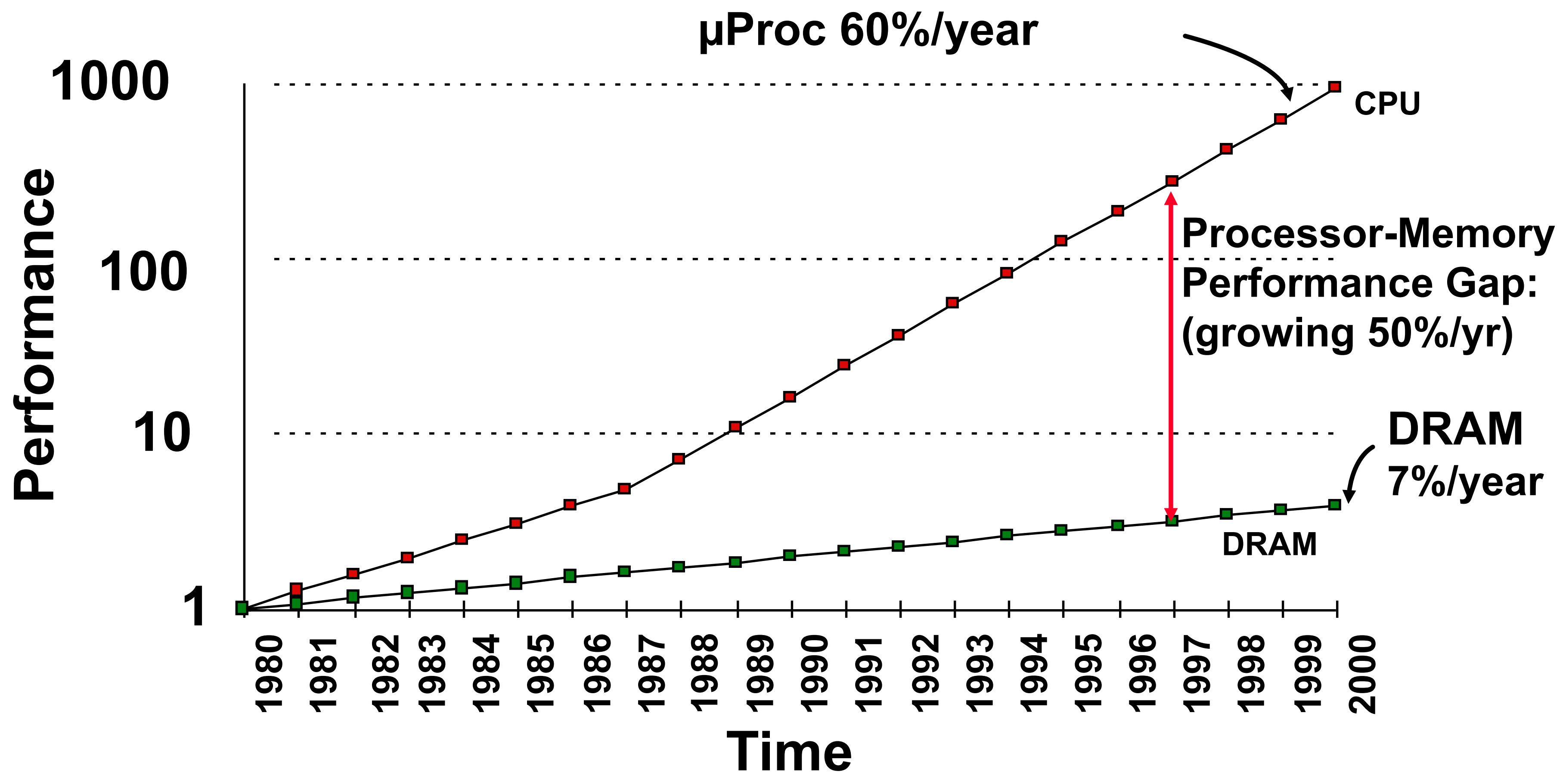
- Design for *Moore's Law*
- Use *abstraction* to simplify design
- Make the *common case fast*
- Performance *via parallelism*
- Performance *via pipelining*
- Performance *via prediction*
- *Hierarchy* of memories
- *Dependability via* redundancy

CPU-Memory Bottleneck



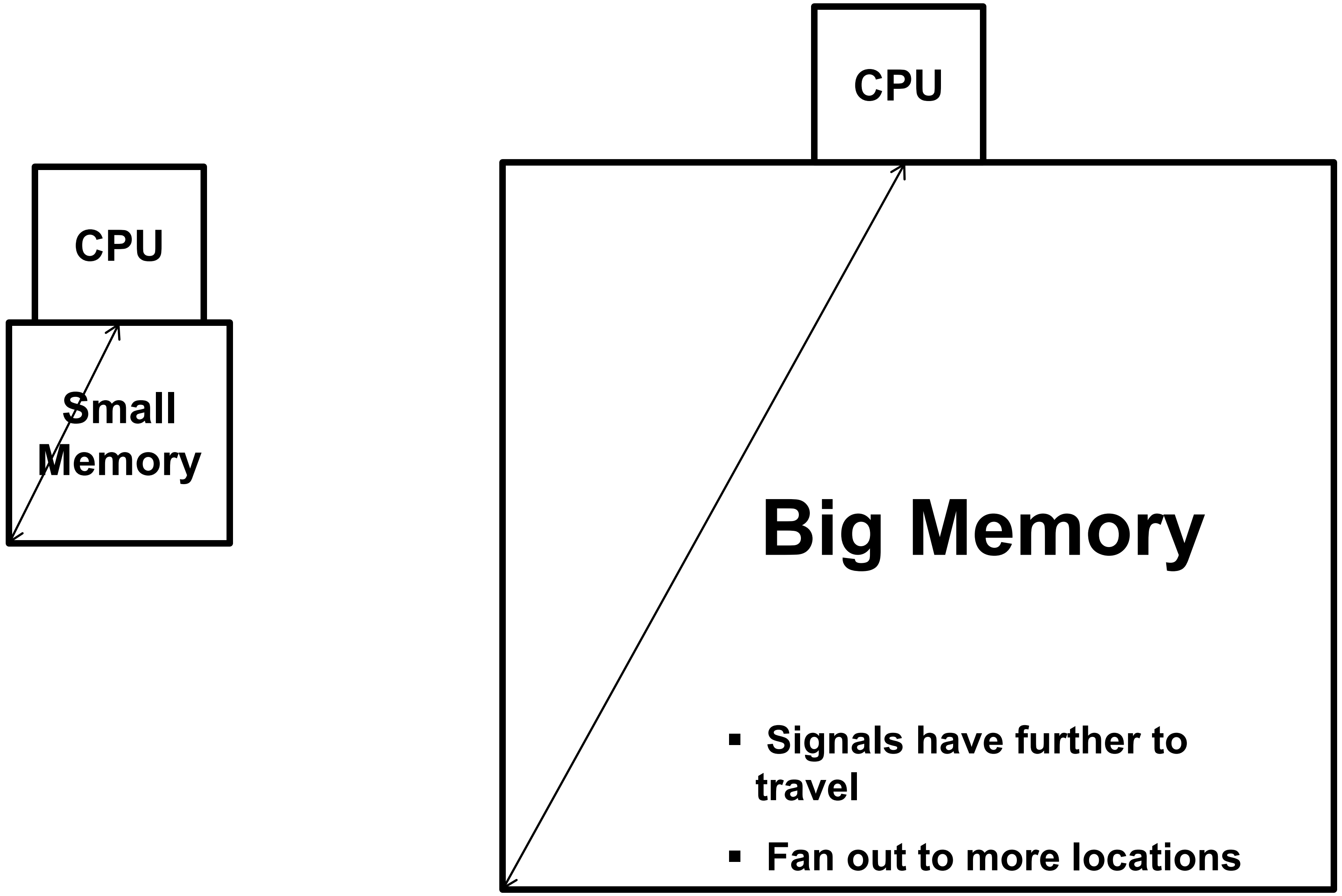
- **Performance of high-speed computers is usually limited by memory bandwidth & latency**
- **Latency (time for a single access)**
  - Memory access time >> Processor cycle time
- **Bandwidth (number of accesses per unit time)**
- **If fraction m of instructions access memory**
  - 1+m memory references / instruction
  - CPI = 1 requires 1+m memory refs / cycle (assuming RISC ISA)
- ***Also, Occupancy (time a memory bank is busy with one request)***

Processor-DRAM Gap (latency)



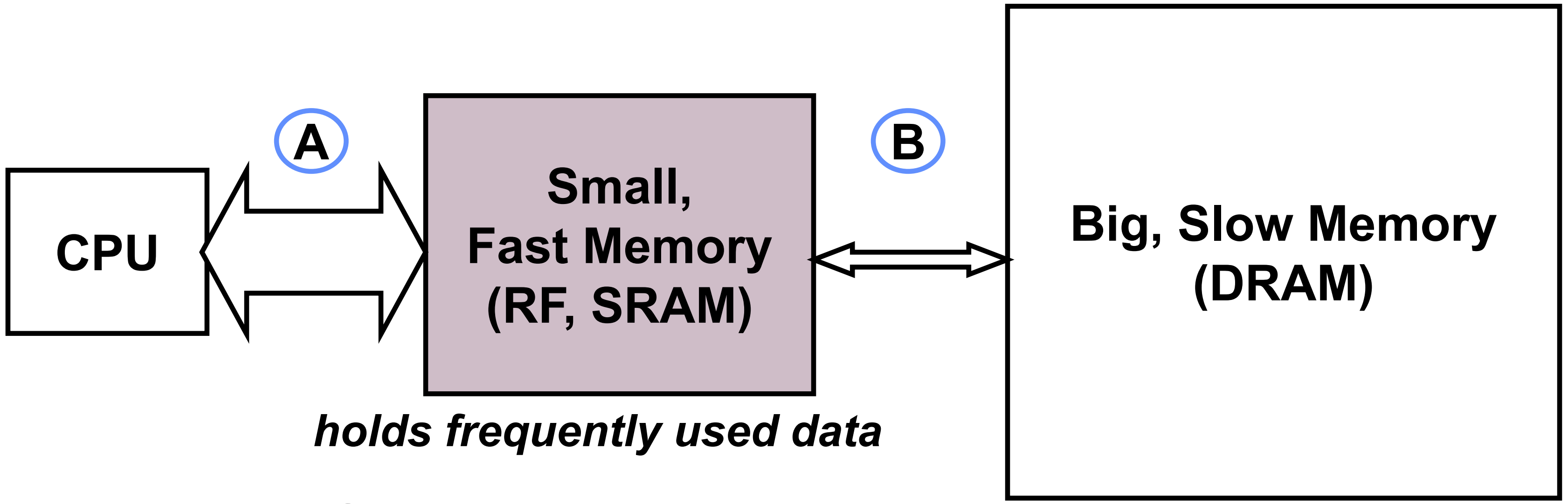
Four-issue 3GHz superscalar accessing 100ns DRAM could execute 1,200 instructions during time for one memory access!

Physical Size Affects Latency



- Signals have further to travel
- Fan out to more locations

Memory Hierarchy



holds frequently used data

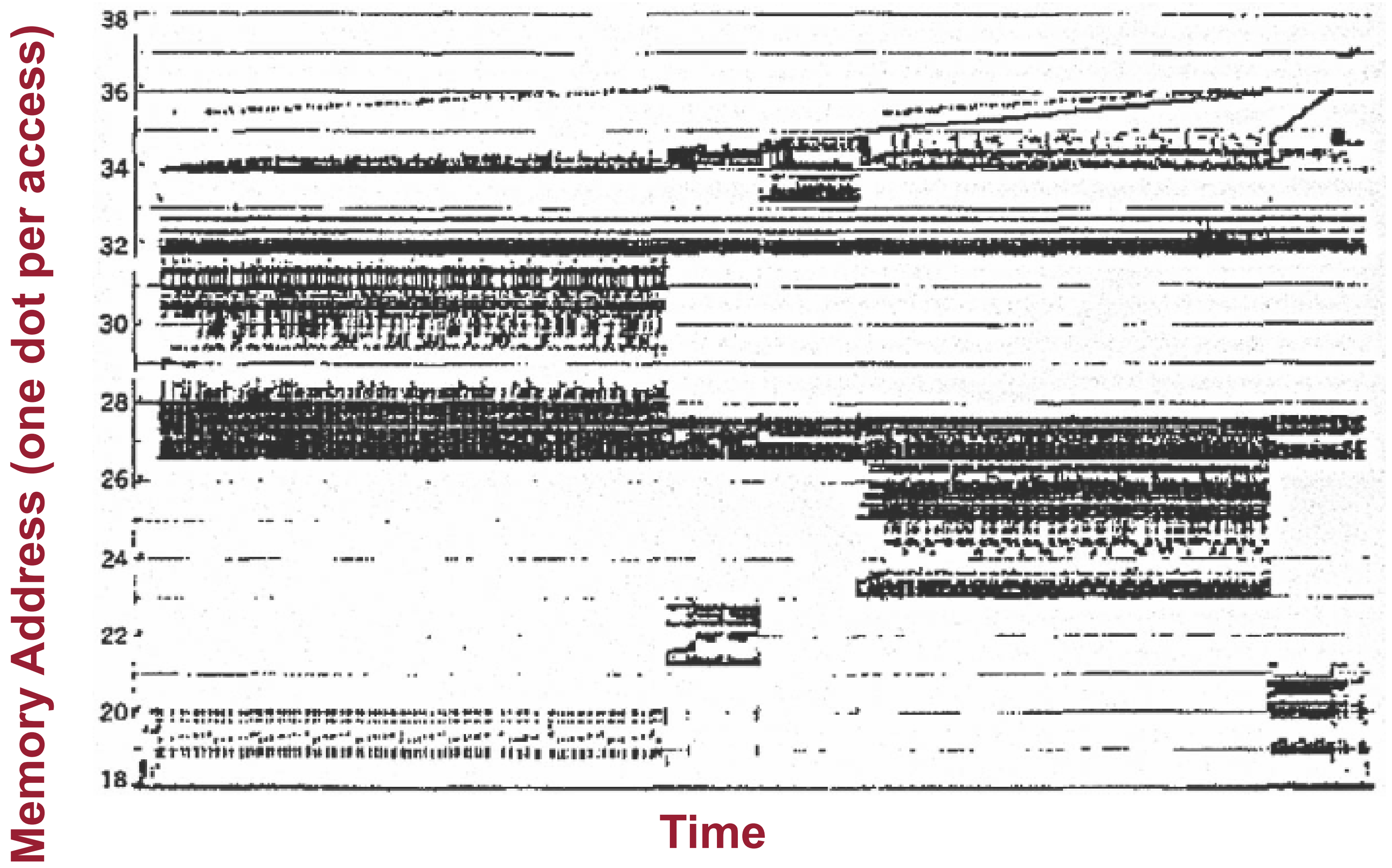
- **Capacity: Register << SRAM << DRAM**
- **Latency: Register << SRAM << DRAM**
- **Bandwidth: on-chip >> off-chip**
- **On a data access**
  - if data $\in$ fast memory $\Longrightarrow$ low latency access (SRAM)
  - if data $\notin$ fast memory $\Longrightarrow$ high latency access (DRAM)

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*

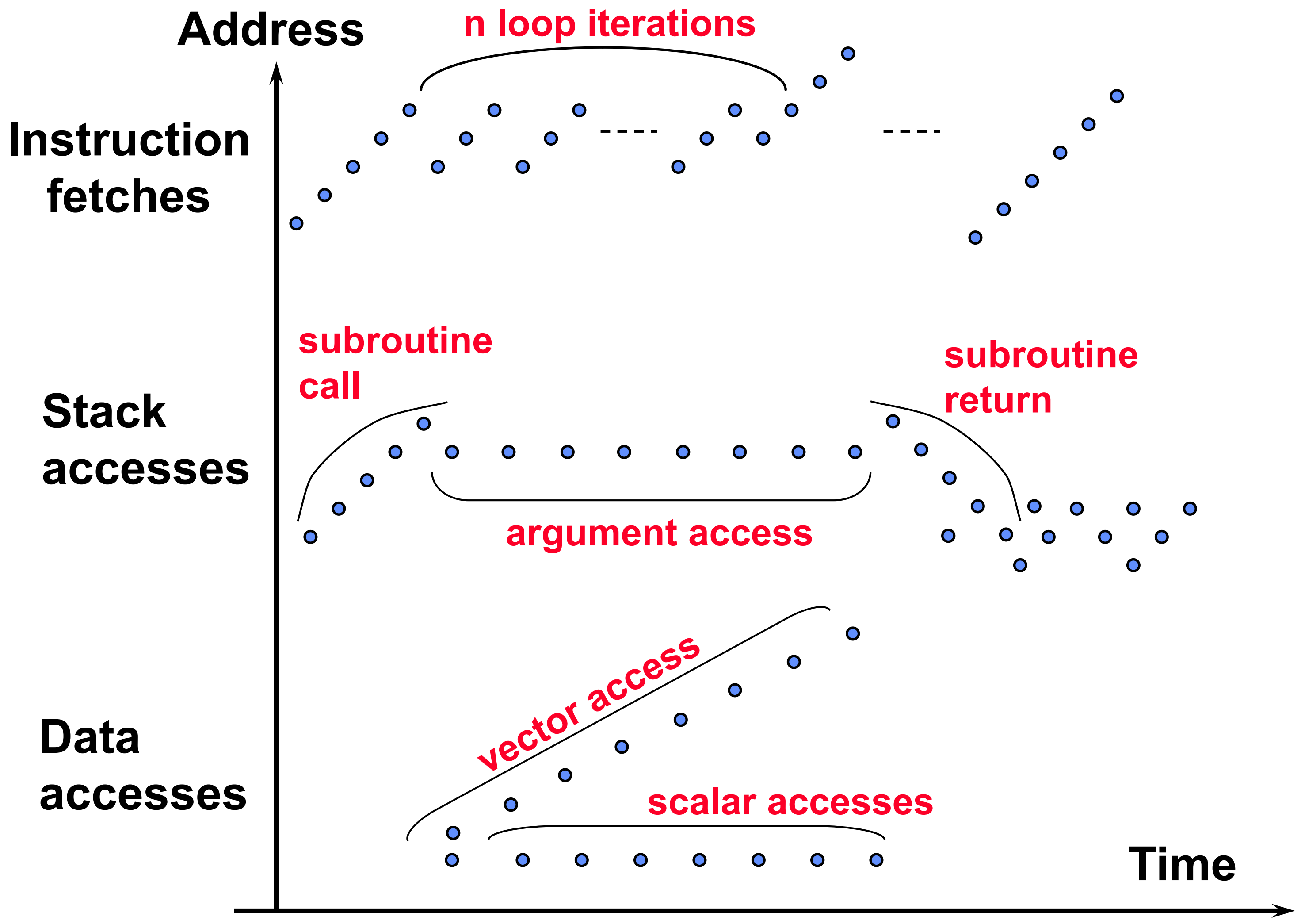
- **Larger/slower storage, e.g., main memory**
  - Address usually computed from values in register
  - Generally implemented as a hardware-managed cache hierarchy (hardware decides what is kept in fast memory)
    - » *But software may provide "hints", e.g., don't cache or prefetch*

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

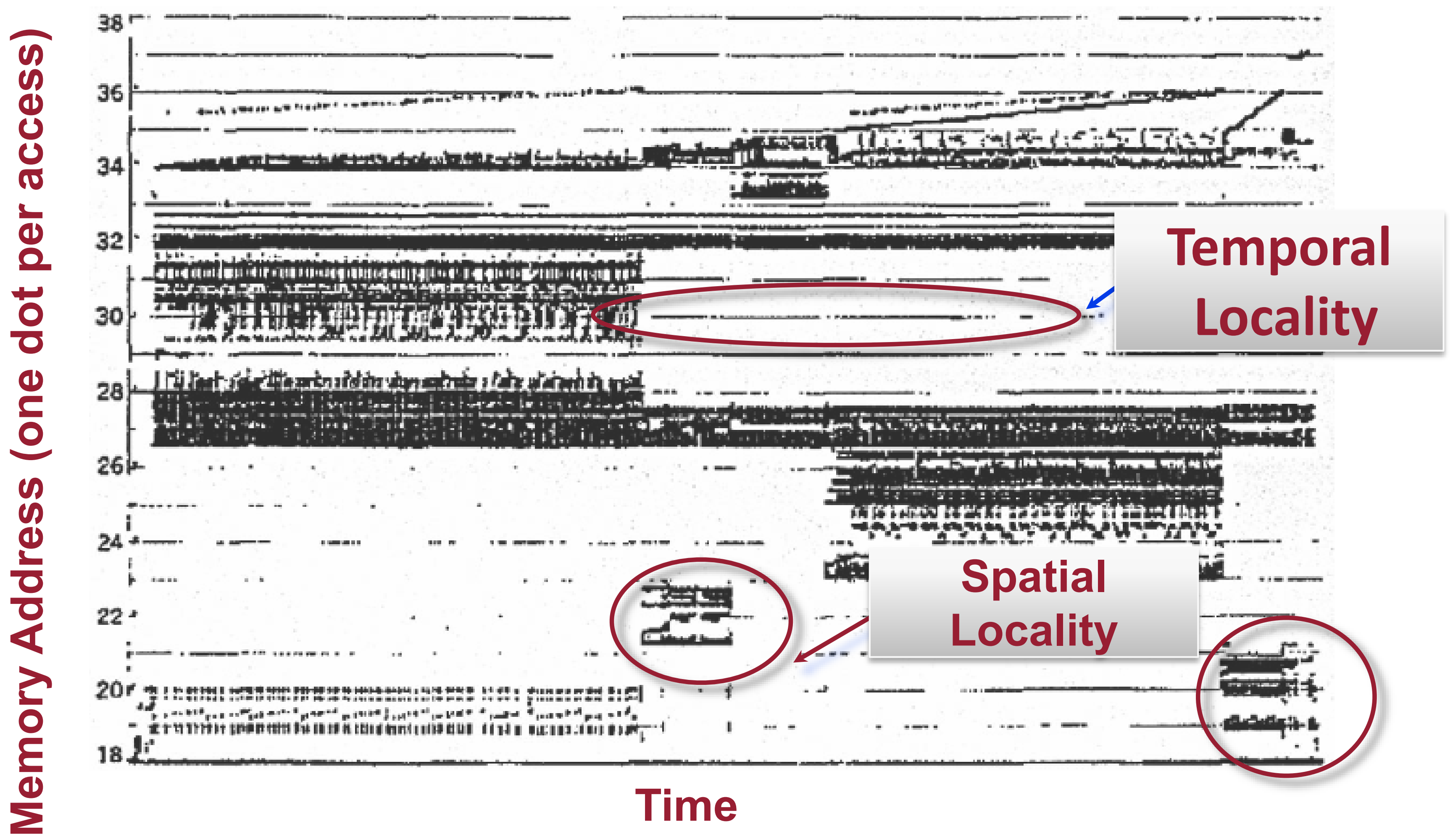


Two Predictable Properties of Memory References

- 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



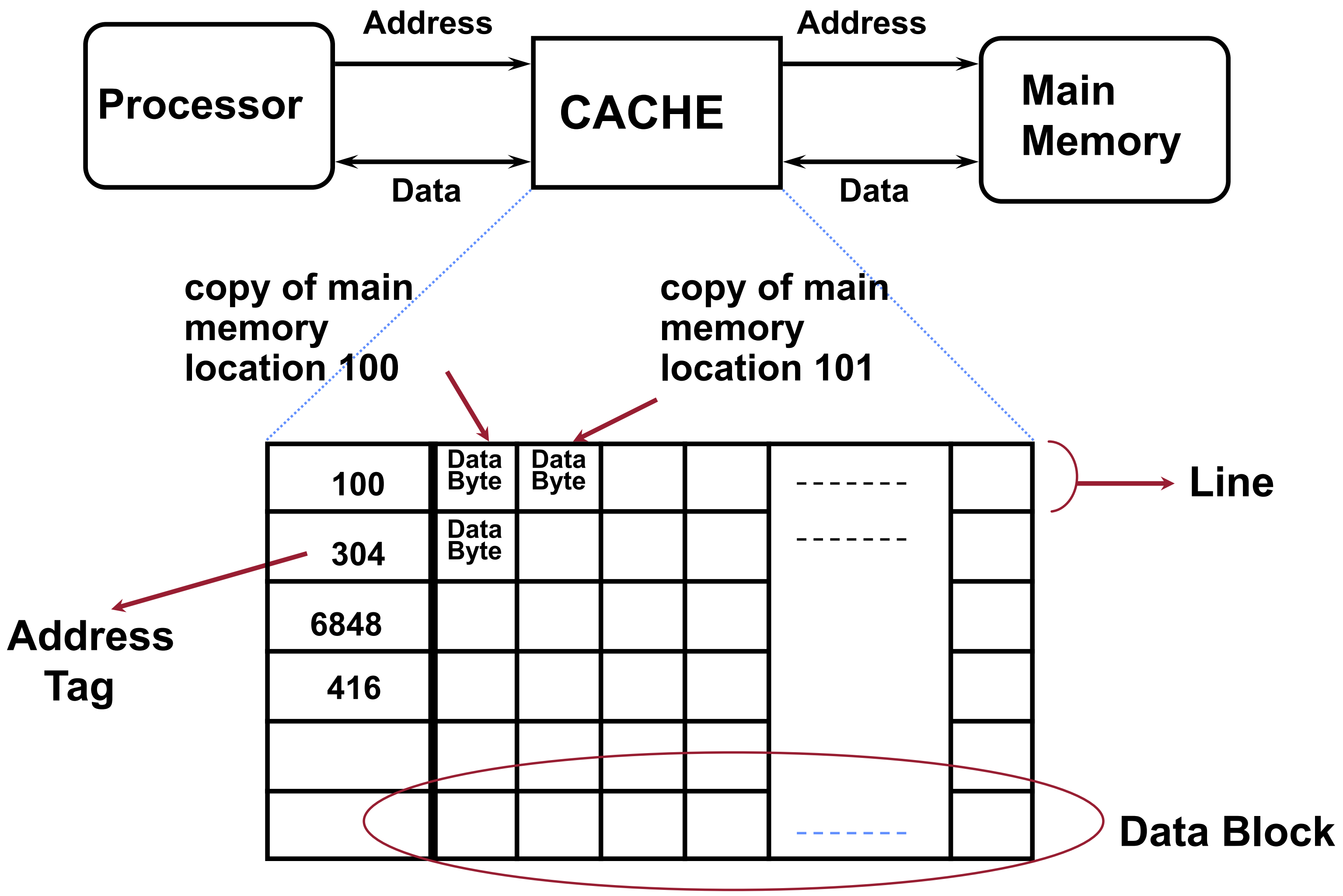
Donald J. Hatfield, Jeanette Gerald: Program Restructuring for Virtual Memory. IBM Systems Journal 10(3): 168-192 (1971)

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

Found in cache
a.k.a. HIT

Return copy of data from cache

Not in cache
a.k.a. MISS

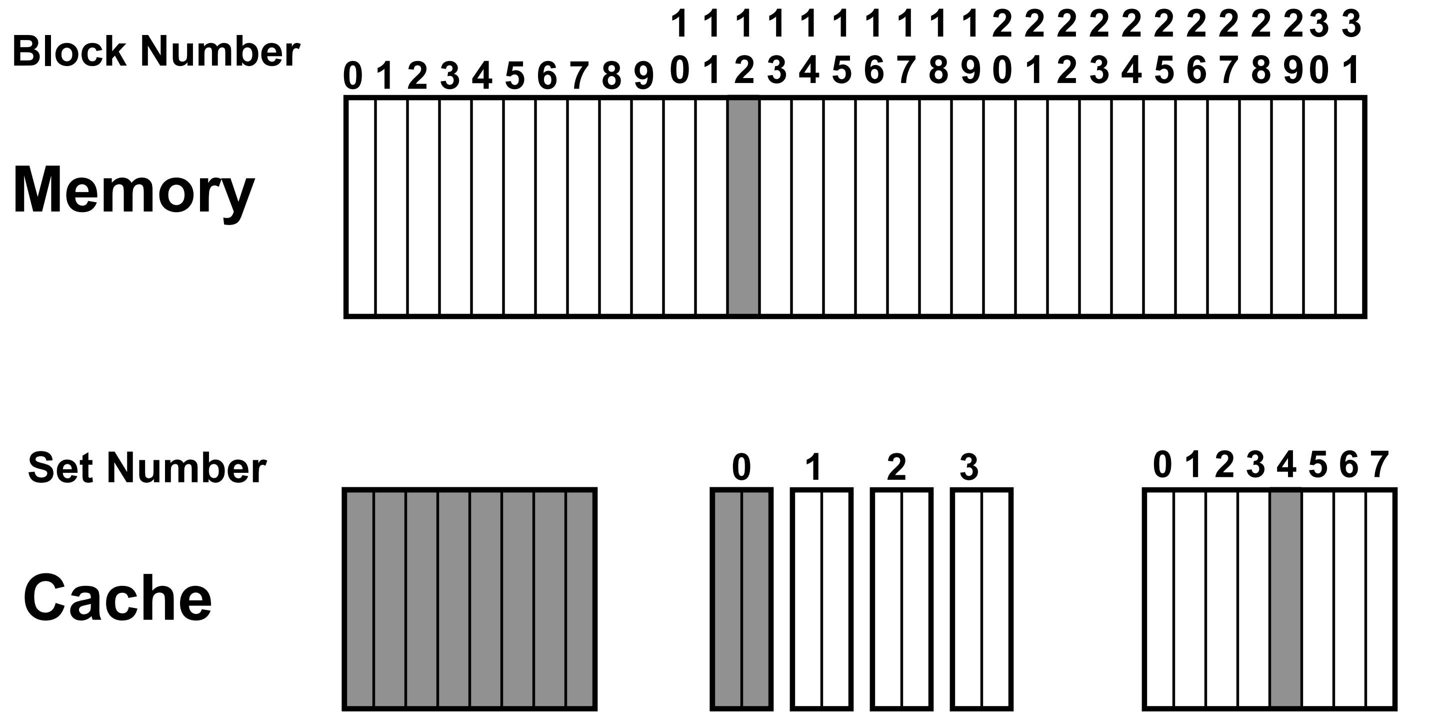
Read block of data from Main Memory

Wait …

Return data to processor and update cache

Q: Which line do we replace?

Placement Policy



	Fully Associative	(2-way) Set Associative	Direct Mapped
block 12 can be placed	anywhere	anywhere in set 0 *(12 mod 4)*	only into block 4 *(12 mod 8)*