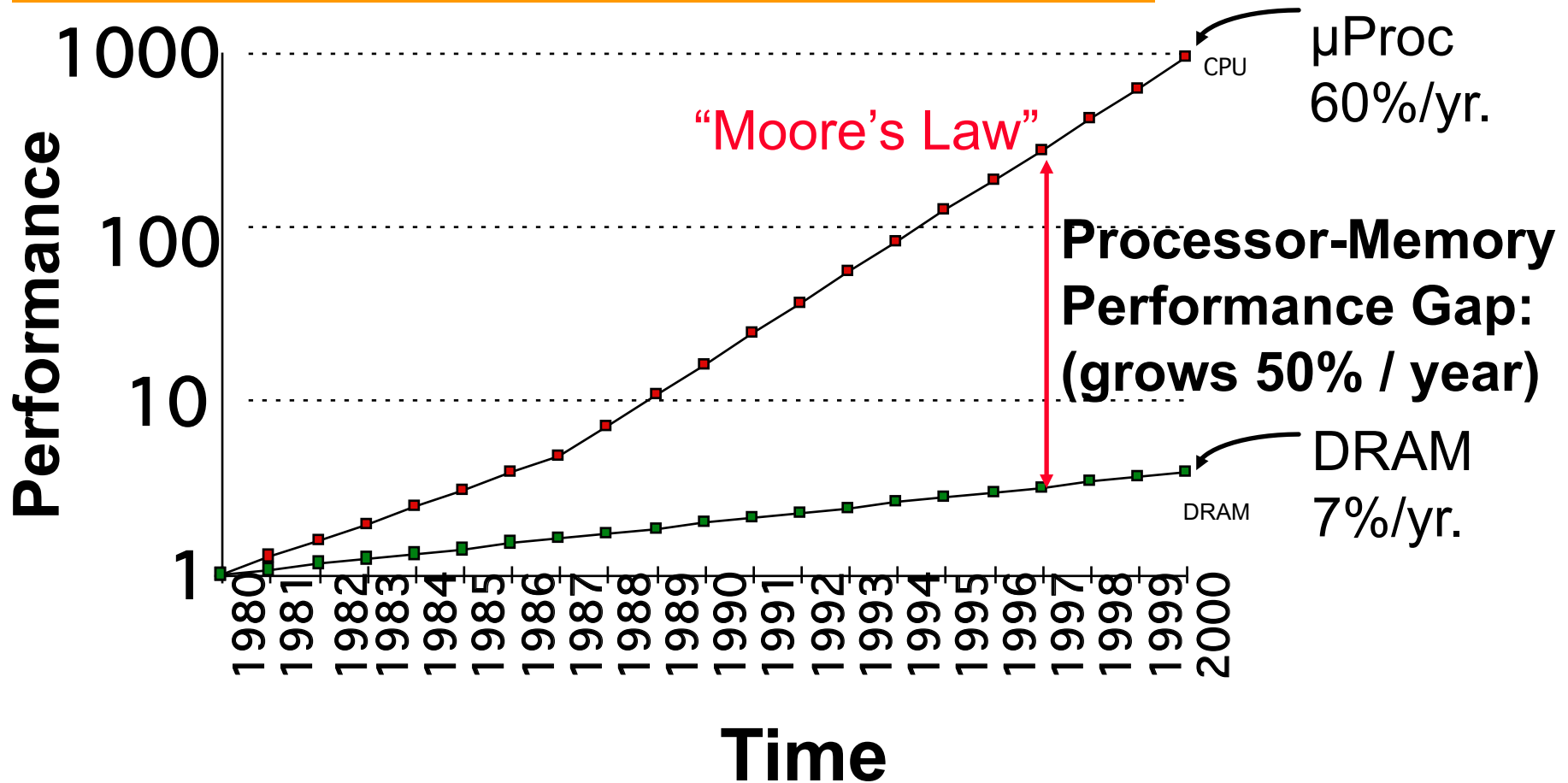


Processor-DRAM Performance Gap

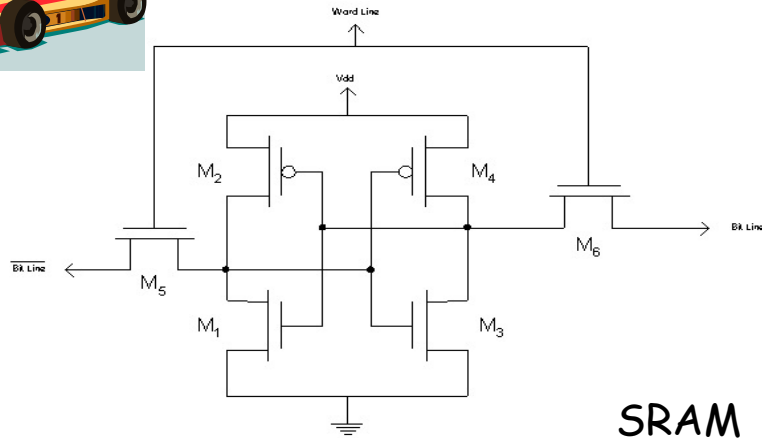
Memory Wall:

1 GHz Processor → 1 ns per clock cycle but 50 ns to go to DRAM
50 processor clock cycles per memory access !!

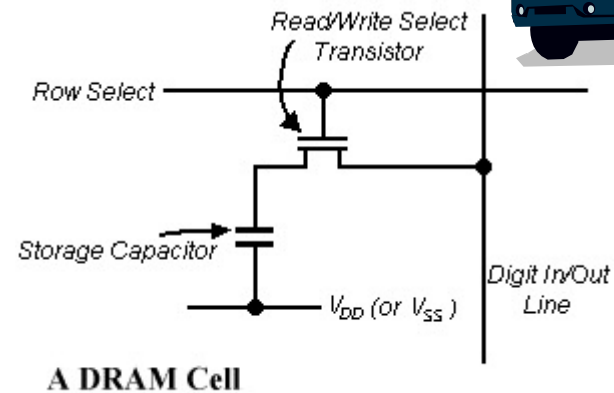


Do we have any faster memory technology?

Fast Memory Technologies: SRAM



SRAM



A DRAM Cell

➤ SRAM

6 transistor per memory cell →

Low density

Fast access latency of 0.5 - 5 ns

➤ DRAM

1 transistor per memory cell →

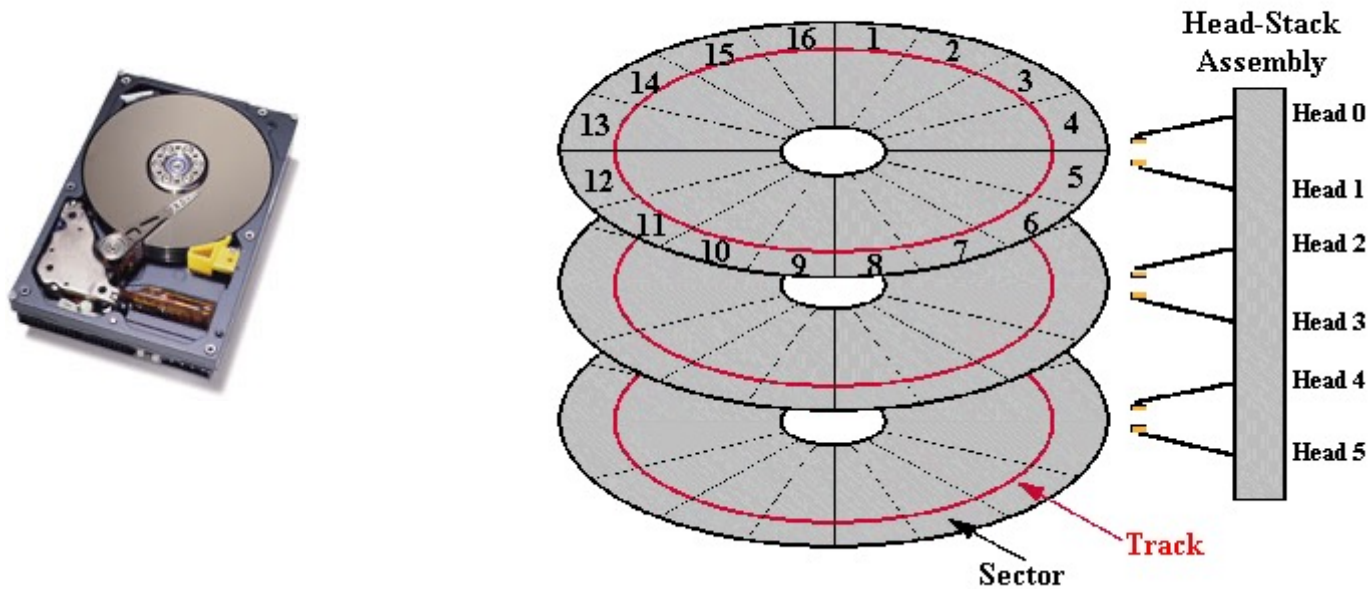
High density

Slow access latency of 50-70ns

Slow Memory Technologies: Magnetic Disk



Drive Physical and Logical Organization

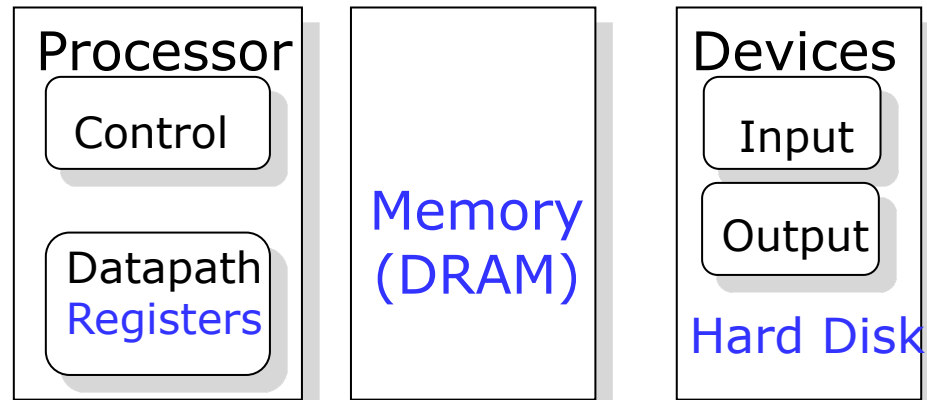


Typical high-end hard disk:

Average latency: 5-20 ms

Capacity: 250GB

Quality vs. Quantity

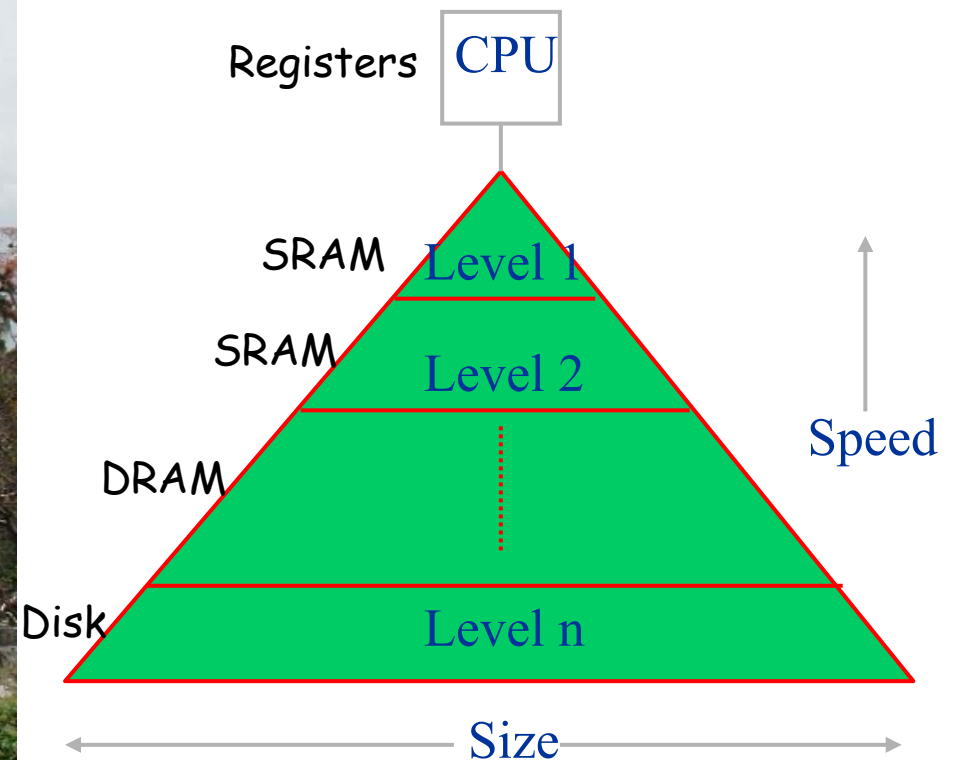


	Capacity	Latency	Cost/GB
Register	100s Bytes	20 ps	\$\$\$\$
SRAM	100s KB	0.5-5 ns	\$\$\$
DRAM	100s MB	50-70 ns	\$
Hard Disk	100s GB	5-20 ms	Cents
Ideal	1 GB	1 ns	Cheap

Best of Both Worlds

- What we want: A BIG and FAST memory
- Memory system should perform like 1GB of SRAM (1ns access time) but cost like 1GB of slow memory
- Key concept: Use a hierarchy of memory technologies
 - Small but fast memory near CPU
 - Large but slow memory farther away from CPU

Memory Hierarchy

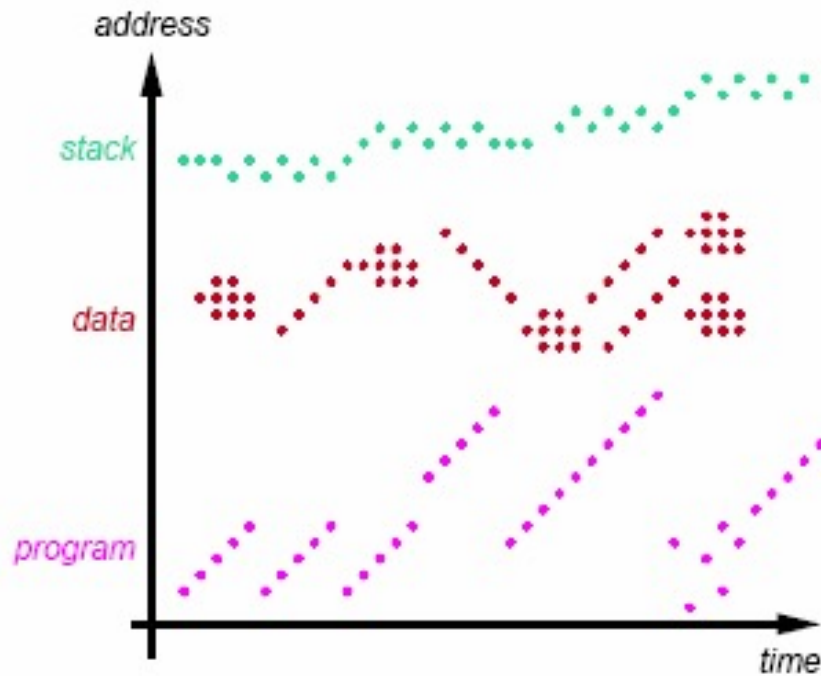


The Basic Idea

- Keep the frequently and recently used data in smaller but faster memory
- Refer to bigger and slower memory only when you cannot find data/instruction in the faster memory
- Why does it work? Principle of Locality

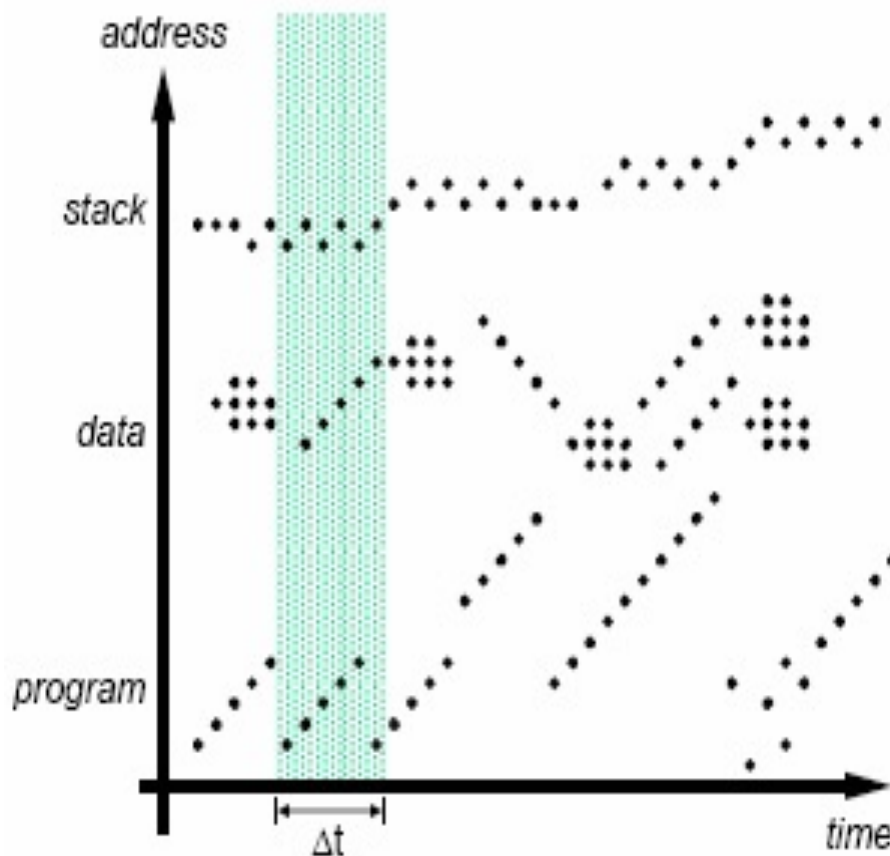
Program accesses only a small portion of the memory address space within a small time interval

Locality



- Temporal locality
 - If an item is referenced, it will tend to be referenced again soon
- Spatial locality
 - If an item is referenced, nearby items will tend to be referenced soon
- Locality for instruction
- Locality for data

Working Set

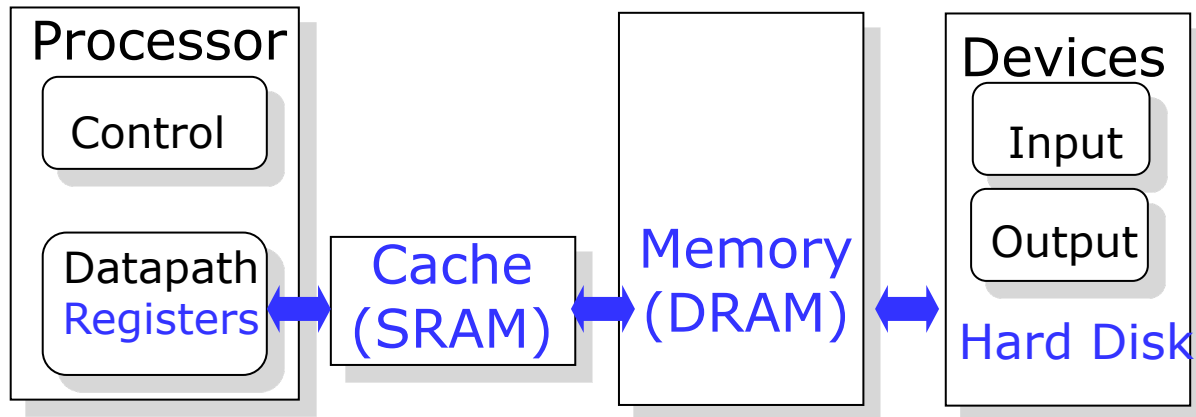


- Set of locations accessed during Δt
- Different phases of execution may use different working sets
- Want to capture the working set and keep it in the memory closest to CPU

Exploiting Memory Hierarchy (1/2)

- Visible to Programmer (Cray etc.)
 - Various storage alternatives, e.g., register, main memory, hard disk
 - Tell programmer to use them cleverly
- Transparent to Programmer (except for performance)
 - Single address space for programmer
 - Processor automatically assigns locations to fast or slow memory depending on usage patterns

Exploiting Memory Hierarchy (2/2)



- How to make SLOW main memory appear faster?
Cache – a small but fast SRAM near CPU
 - Often in the same chip as CPU
 - Introduced in 1960; almost every processor uses it today
 - Hardware managed: Transparent to programmer
- How to make SMALL main memory appear bigger than it is? Virtual memory
 - OS managed: Again transparent to programmer
- What about registers?