CMSC 611: Advanced Computer Architecture

Memory, I/O and Disk

Most slides adapted from David Patterson. Some from Mohomed Younis

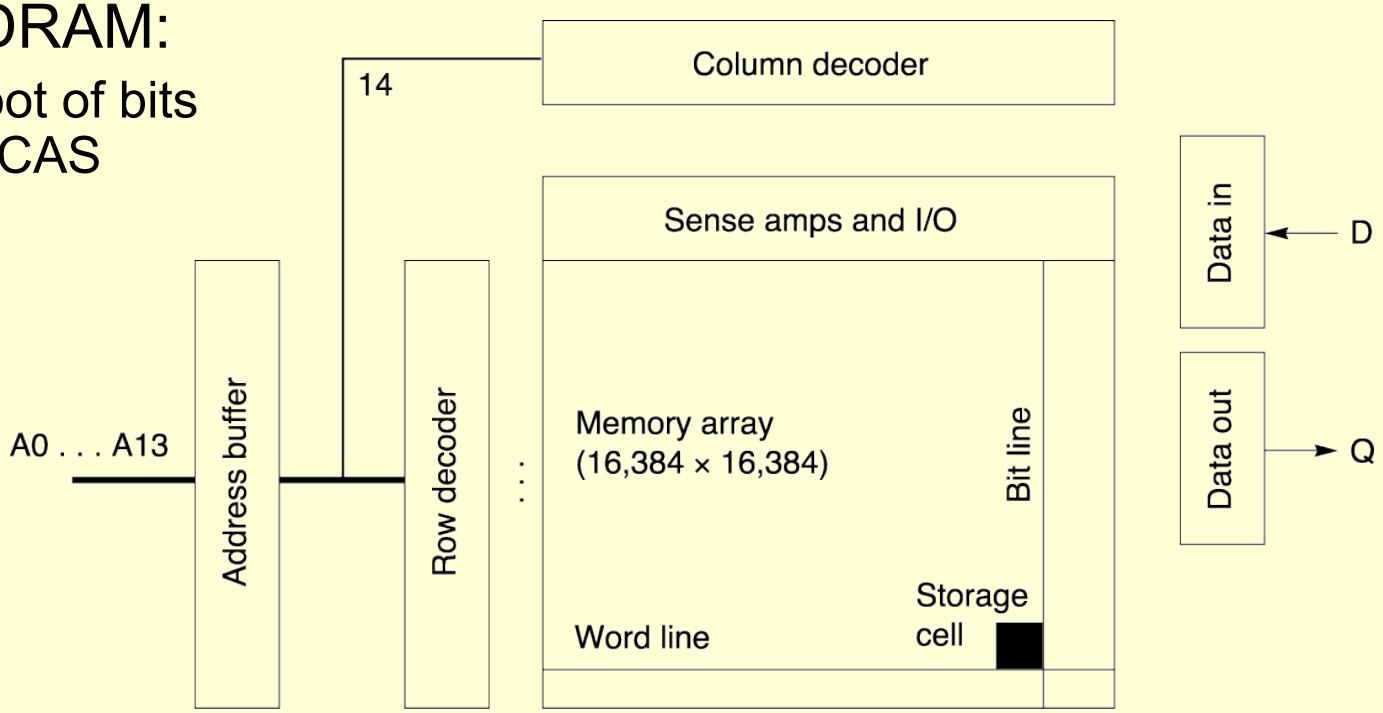
Main Memory Background

- Performance of Main Memory:
  - Latency: affects cache miss penalty
    - Access Time: time between request and word arrives
    - Cycle Time: time between requests
  - Bandwidth: primary concern for I/O & large block
- Main Memory is DRAM: Dynamic RAM
  - Dynamic since needs to be refreshed periodically
  - Addresses divided into 2 halves (Row/Column)
- Cache uses SRAM: Static RAM
  - No refresh
    - 6 transistors/bit vs. 1 transistor/bit, 10X area
  - Address not divided: Full address

DRAM Logical Organization

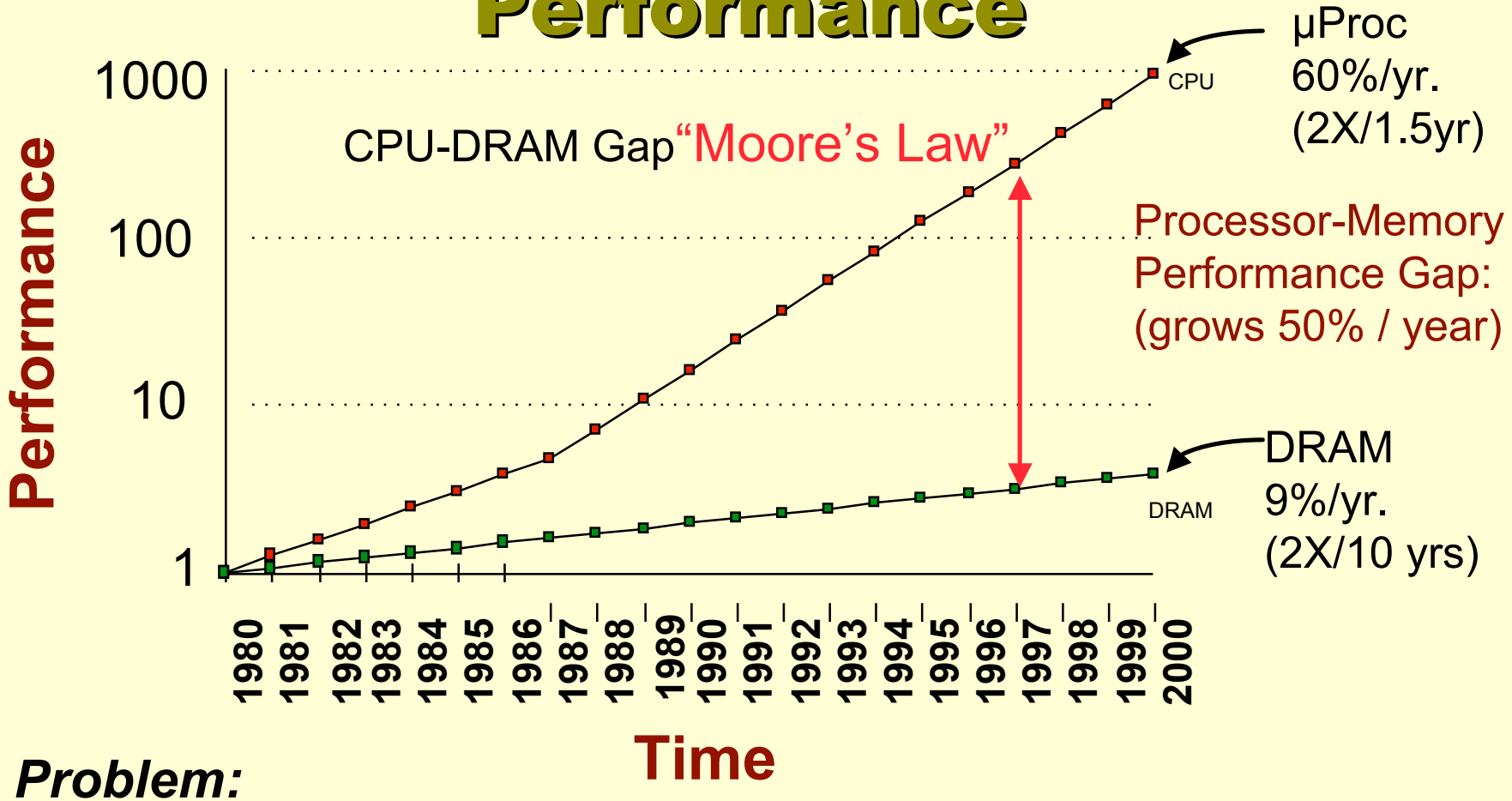
4 Mbit DRAM:

square root of bits per RAS/CAS



- Refreshing prevent access to the DRAM (typically 1-5% of the time)
- Reading one byte refreshes the entire row
- Read is destructive and thus data need to be re-written after reading
  - Cycle time is significantly larger than access time

Processor-Memory Performance



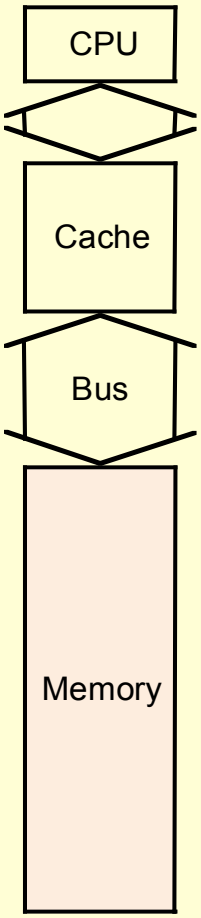
Problem:

Improvements in access time are not enough to catch up

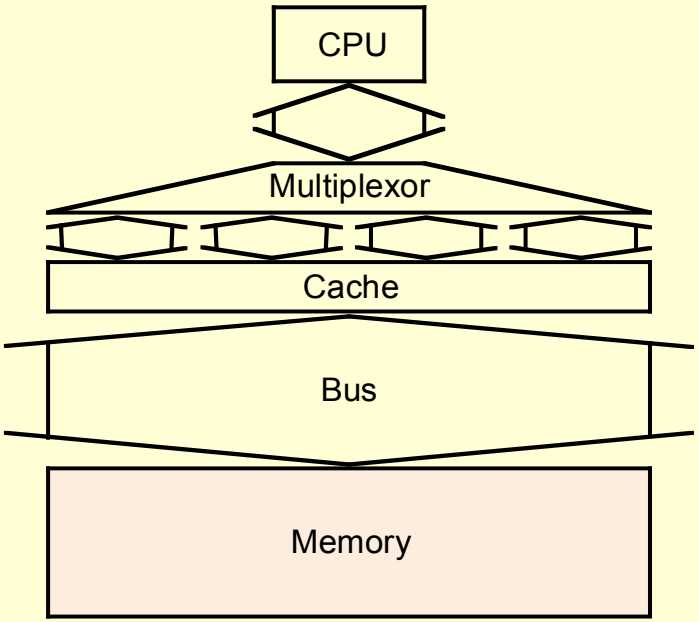
Solution:

Increase the bandwidth of main memory (improve throughput)

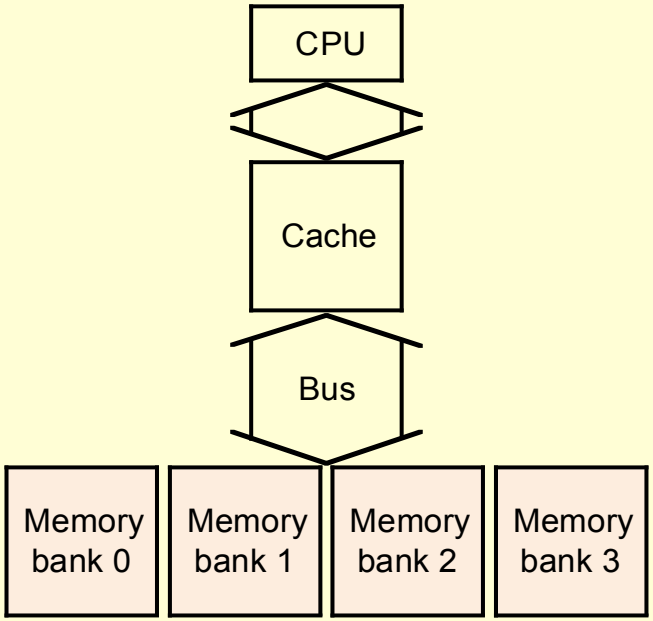
Memory Organization



a. One-word-wide memory organization



b. Wide memory organization



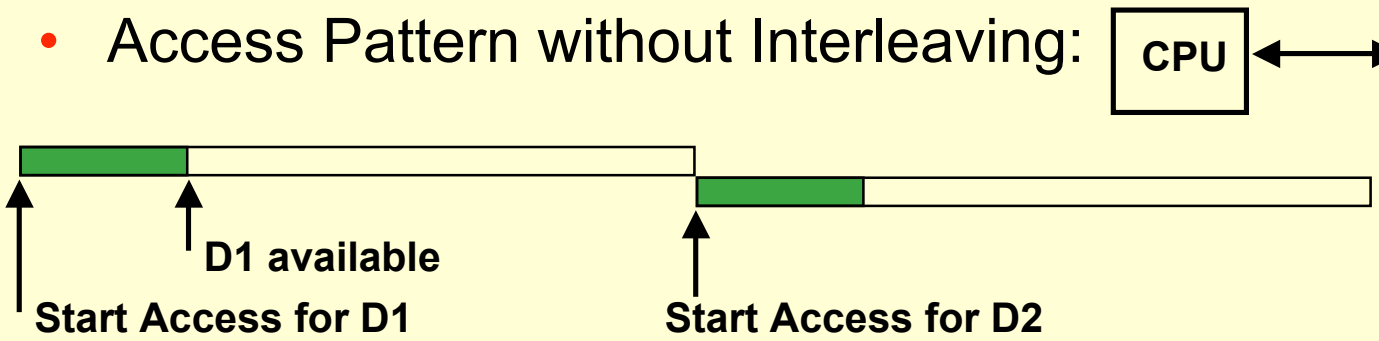
c. Interleaved memory organization

- *Simple*: CPU, Cache, Bus, Memory same width (32 bits)
- *Wide*: CPU/Mux 1 word; Mux/Cache, Bus, Memory N words
- *Interleaved*: CPU, Cache, Bus 1 word: Memory N Modules (4 Modules); example is *word interleaved*

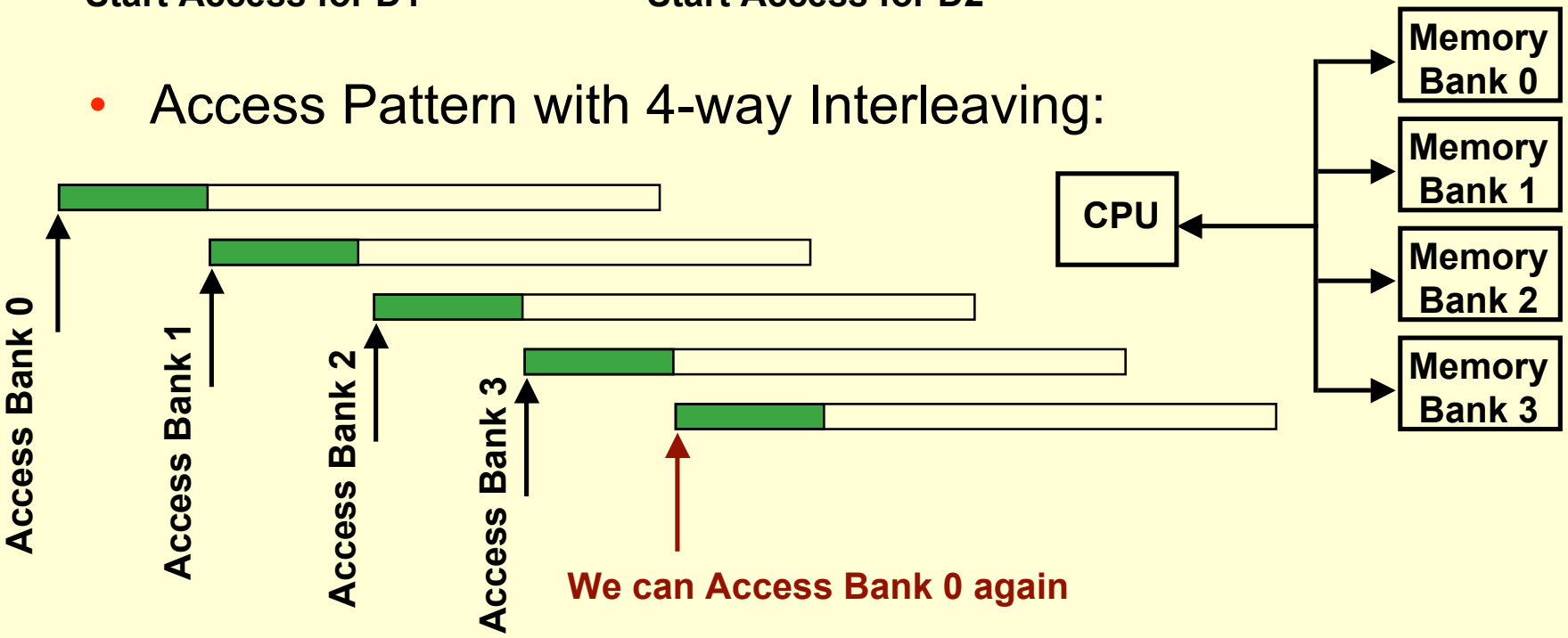
Memory organization would have significant effect on bandwidth

Memory Interleaving

- Access Pattern without Interleaving:

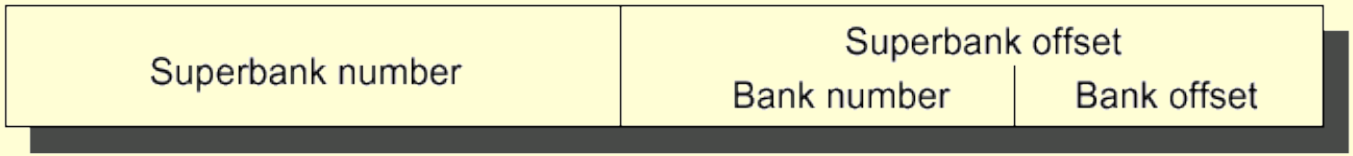


- Access Pattern with 4-way Interleaving:



Independent Memory Banks

- Original motivation: sequential access
- Multiple independent accesses:
  - Multiprocessor system / concurrent execution
  - I/O: limiting memory access contention
  - CPU with Hit under n Misses, Non-blocking Cache
- Multiple access requires per-bank
  - controller, address bus and possibly data bus



  - Superbank: all memory active on one block transfer
  - Bank: portion within a superbank that is word interleaved (or Subbank)

Superbanks act as separate memories mapped to the same address space

Avoiding Bank Conflicts

- The effectiveness of interleaving depends on the frequency that independent requests will go to different banks
  - Bank number = address % number of banks
  - Address within bank = address / words in bank
- Example: Assuming 128 banks

```
int x[256][512];
for (j = 0; j < 512; j = j+1)
    for (i = 0; i < 256; i = i+1)
          x[i][j] = 2 * x[i][j];
```

- Since 512 is a multiple of 128
  - Every column in same bank

Avoiding Bank Conflicts

- Solutions:
  - SW: loop interchange or declaring array not power of 2 (“array padding”)
  - HW: Prime number of banks and modulo interleaving
    - Complexity of modulo & divide per memory access with prime no. banks?
    - Simple address calculation using the *Chinese Remainder Theorem*

Chinese Remainder Theorem

- As long as two sets of integers a_i and b_i follow these rules:
  - $b_i = x \bmod a_i$
  - $0 \leq b_i < a_i$
  - $0 \leq x < a_0 \times a_1 \times a_2 \times \ldots$
  - a_i and a_j are co-prime with $i \neq j$
- then the integer x has only one solution

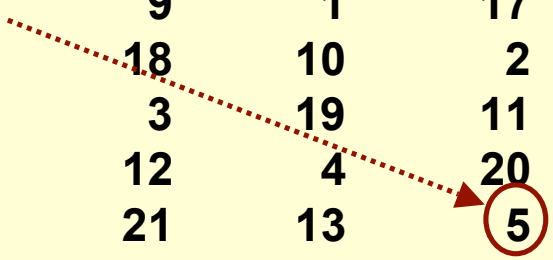
Apply to Bank Addressing

- Modulo interleaving
  - Bank number = b_0, Number of banks = a_0
  - Address in bank = b_1, Words in bank = a_1
  - Bank number < number of banks ($0 \leq b_0 < a_0$)
  - Address in bank < words in a bank ($0 \leq b_1 < a_1$)
  - Address < Number of banks × words in bank
    - ($0 \leq x < a_0 \times a_1$)
  - The number of banks (a_0) and words in a bank are co-prime (a_1)
    - a_1 a power of 2, a_0 prime > 2

Example

- Bank number = address MOD number of banks
- Address within bank = address MOD words in bank
- Bank number = b_0, number of banks = a_0 (= 3 in example)
- Address within bank = b_1, number of words in bank = a_1 (= 8 in example)

	Seq. Interleaved			Modulo Interleaved		
Bank Number:	**0**	**1**	**2**	**0**	**1**	**2**
Address within Bank:						
0	0	1	2	0	16	8
1	3	4	5	9	1	17
2	6	7	8	18	10	2
3	9	10	11	3	19	11
4	12	13	14	12	4	20
5	15	16	17	21	13	5
6	18	19	20	6	22	14
7	21	22	23	15	7	23



Unambiguous mapping with simple bank addressing

DRAM-specific Optimization

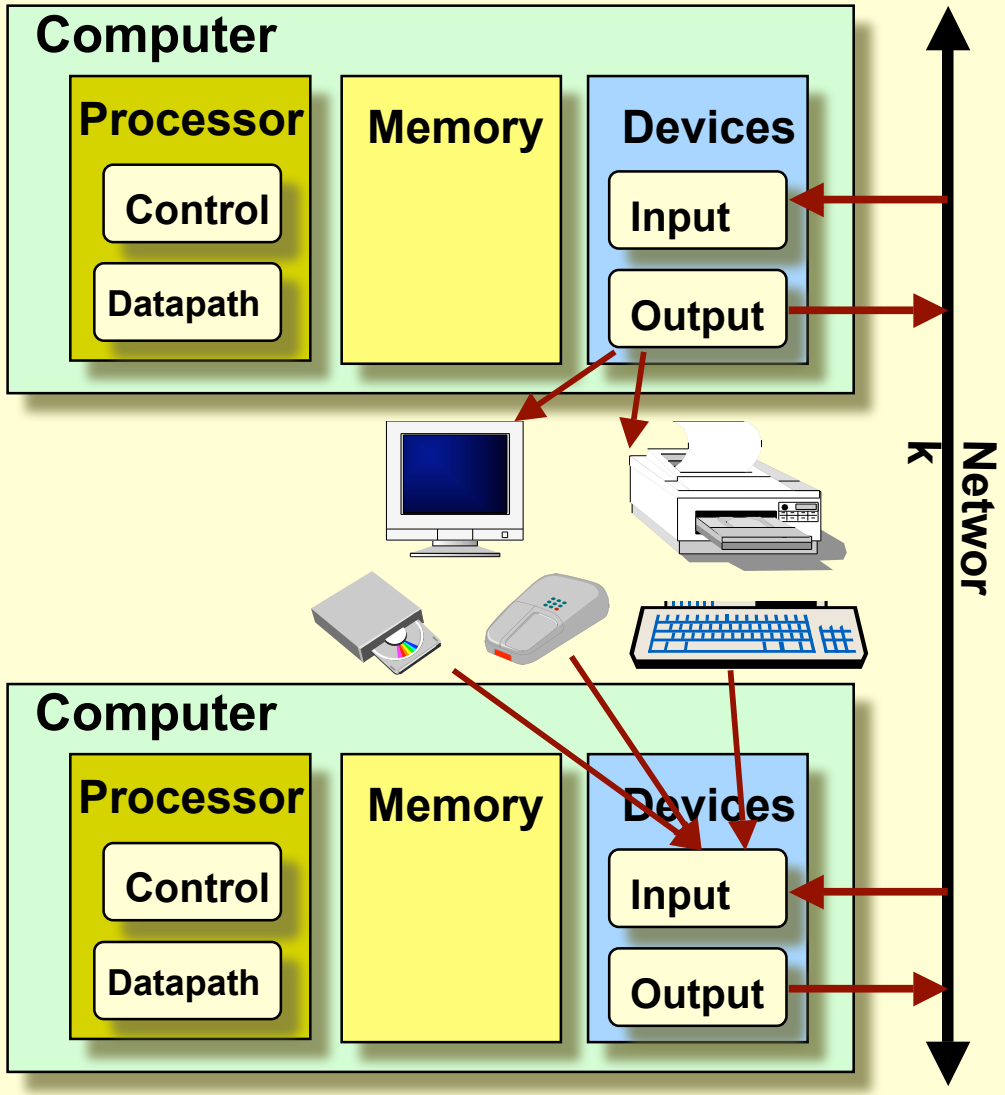
- DRAM Access Interleaving
  - DRAM must buffer a row of bits internally for the column access
    - Performance can be improved by allowing repeated access to buffer with our another row access time (requires minimal additional cost)
  - Page mode: The buffer acts like a SRAM allowing bit access from the buffer until a row change or a refresh
  - Static column: Similar to page mode but does not require change in CAS to access another bit from the buffer
  - DRAM optimization has been shown to give up to 4x speedup

DRAM-Specific Optimization

- Bus-based DRAM (RAMBUS)
  - Each chip act as a module with an internal bus replacing CAS and RAS
  - Allows for other accesses to take place between the sending the address and returning the data
  - Each module performs its own refresh
  - Performance can reach 1 byte / 2 ns
    - (500 MB/s per chip)
  - Expensive compared to traditional DRAM

Input/Output

- I/O Interface
  - Device drivers
  - Device controller
  - Service queues
  - Interrupt handling
- Design Issues
  - Performance
  - Expandability
  - Standardization
  - Resilience to failure
- Impact on Tasks
  - Blocking conditions
  - Priority inversion
  - Access ordering



Impact of I/O on System Performance

Suppose we have a benchmark that executes in 100 seconds of elapsed time, where 90 seconds is CPU time and the rest is I/O time. If the CPU time improves by 50% per year for the next five years but I/O time does not improve, how much faster will our program run at the end of the five years?

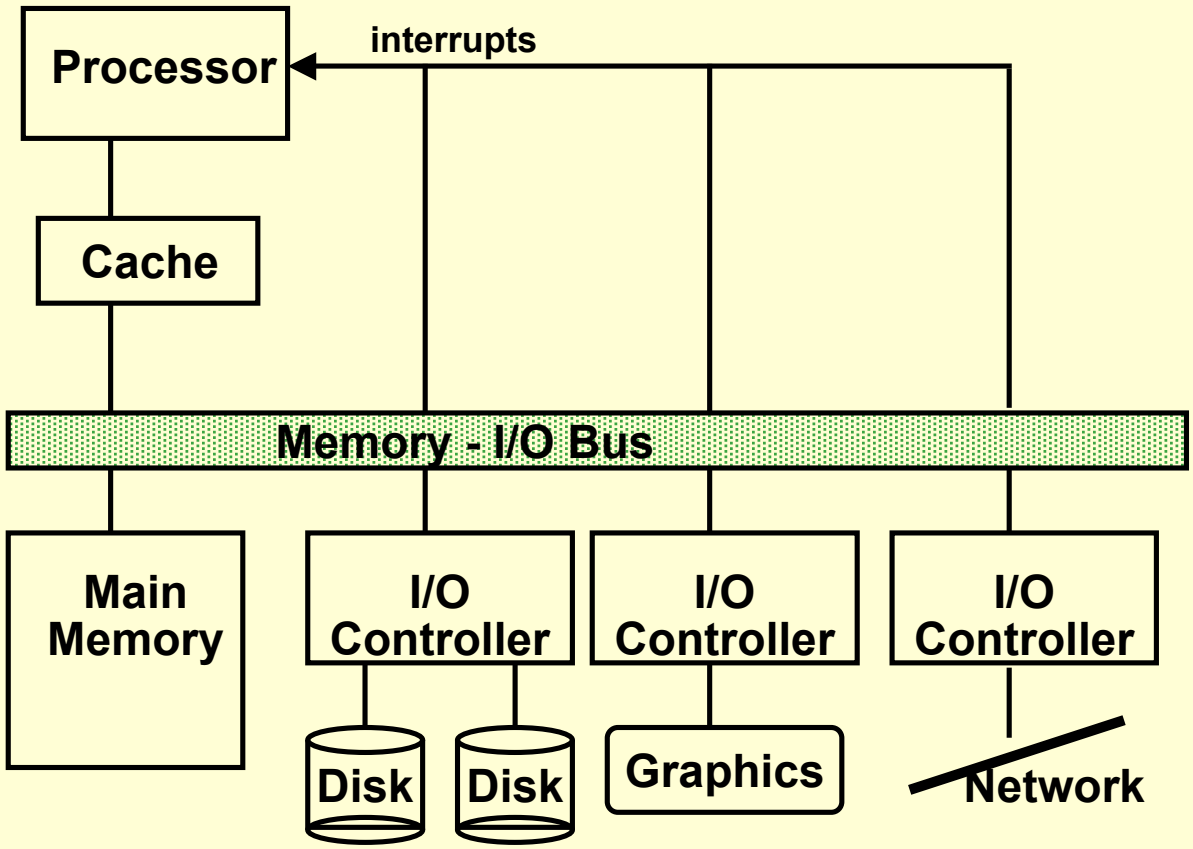
Answer: Elapsed Time = CPU time + I/O time

After n years	CPU time	I/O time	Elapsed time	% I/O time
0	90 Seconds	10 Seconds	100 Seconds	10%
1	$\frac{90}{1.5}$ = 60 Seconds	10 Seconds	70 Seconds	14%
2	$\frac{60}{1.5}$ = 40 Seconds	10 Seconds	50 Seconds	20%
3	$\frac{40}{1.5}$ = 27 Seconds	10 Seconds	37 Seconds	27%
4	$\frac{27}{1.5}$ = 18 Seconds	10 Seconds	28 Seconds	36%
5	$\frac{18}{1.5}$ = 12 Seconds	10 Seconds	22 Seconds	45%

Over five years:

CPU improvement = 90/12 = 7.5 **BUT** System improvement = 100/22 = 4.5

Typical I/O System



- The connection between the I/O devices, processor, and memory are usually called (local or internal) bus
- Communication among the devices and the processor use both protocols on the bus and interrupts

I/O Device Examples

Device	***Behavior***	***Partner***	***Data Rate (KB/sec)***
Keyboard	Input	Human	0.01
Mouse	Input	Human	0.02
Line Printer	Output	Human	1.00
Floppy disk	Storage	Machine	50.00
Laser Printer	Output	Human	100.00
Optical Disk	Storage	Machine	500.00
Magnetic Disk	Storage	Machine	5,000.00
Network-LAN	Input or Output	Machine	20 – 1,000.00
Graphics Display	Output	Human	30,000.00

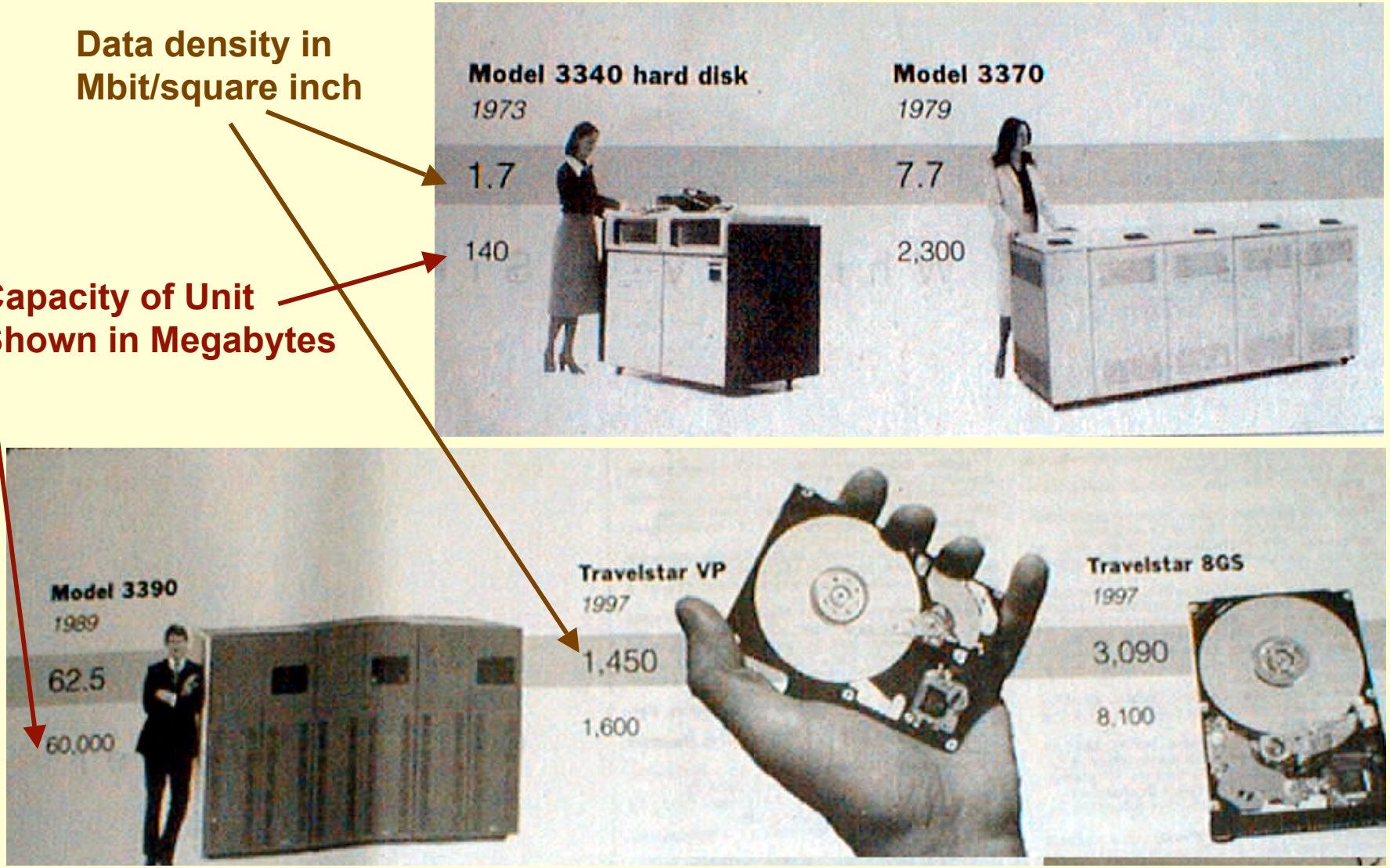
Storage Technology Drivers

- Driven by the prevailing computing paradigm:
  - 1950s: migration from batch to on-line processing
  - 1990s: migration to ubiquitous computing
    - Computers in phones, books, cars, video cameras, …
    - Nationwide fiber optical network with wireless tails
- Effects on storage industry:
  - Embedded storage: smaller, cheaper, more reliable, lower power
  - Data utilities: high capacity, hierarchically managed storage

Disk History

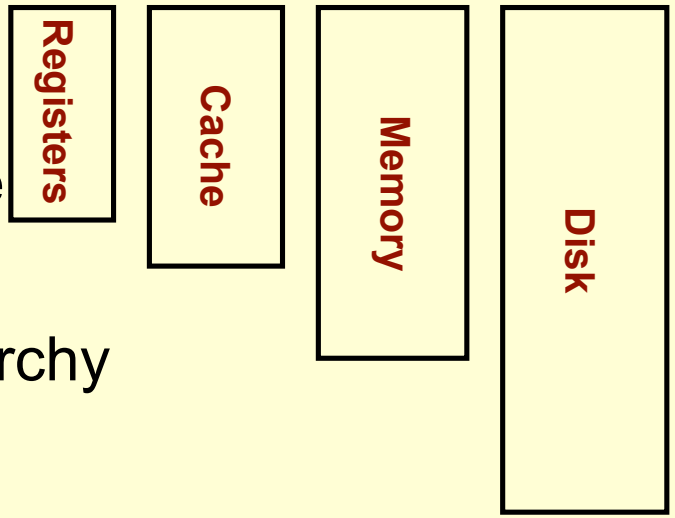
Data density in
Mbit/square inch

Capacity of Unit
Shown in Megabytes



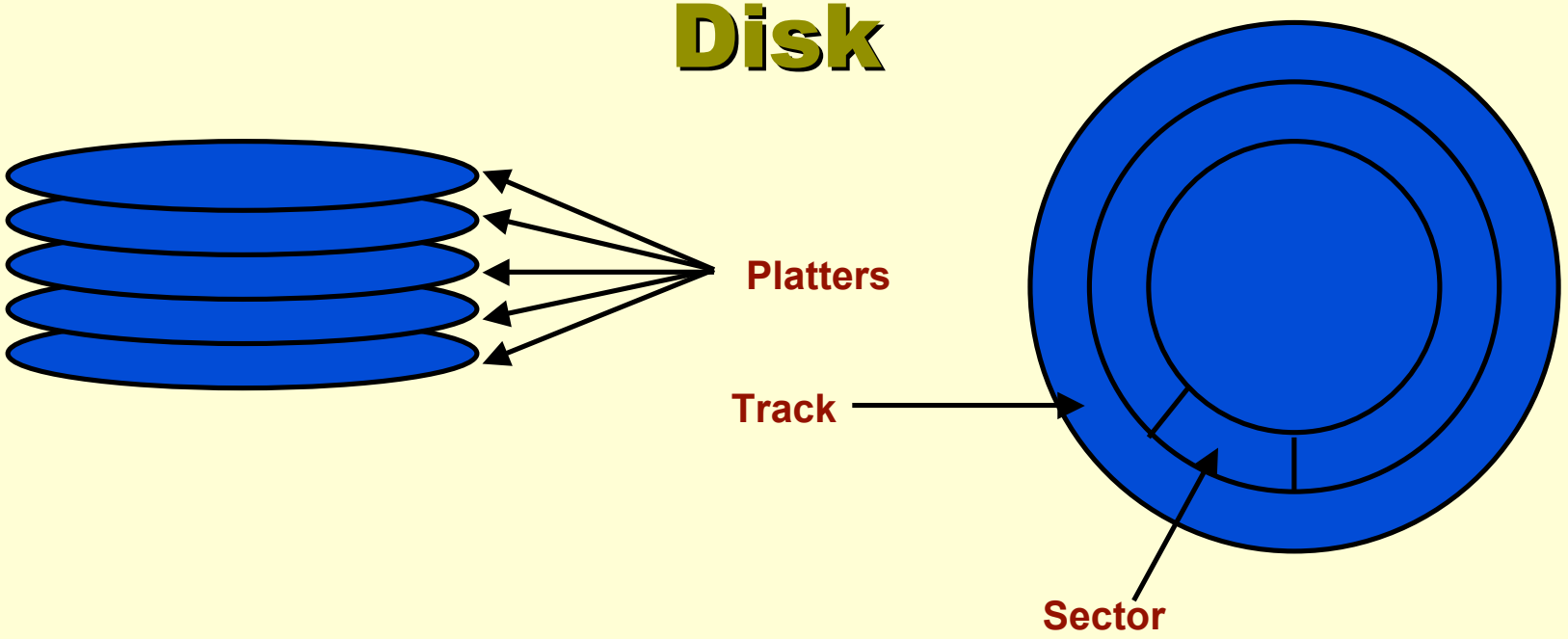
source: New York Times, 2/23/98, page C3

Magnetic Disk



- Purpose:
  - Long term, nonvolatile storage
  - Large, inexpensive, and slow
  - Low level in the memory hierarchy
- Characteristics:
  - Rely on rotating platters coated with a magnetic surface
  - Use a moveable read/write head to access the disk
  - Platters are rigid (metal or glass)

Organization of a Hard Magnetic Disk



- Typical numbers (depending on the disk size):
  - 500 to 2,000 tracks per surface
  - 32 to 128 sectors per track
    - A sector is the smallest unit that can be read or written to
- Traditionally all tracks have the same number of sectors:
  - Constant bit density: record more sectors on the outer tracks
  - Recently relaxed: constant bit size, speed varies with track location