Performance & Technology

Todd C. Mowry
CS 740

Sept 12, 2001

Topics:

- Performance measures
- Relating performance measures
- Memory Technology
  - SRAM, DRAM
- Disk Technology

Performance expressed as a time

Absolute time measures

- difference between start and finish of an operation
- synonyms: running time, elapsed time, response time, latency, completion time, execution time
- most straightforward performance measure

Relative (normalized) time measures

- running time normalized to some reference time
- (e.g. time/reference time)

Guiding principle: Choose performance measures that track running time.

Performance expressed as a rate

Rates are performance measures expressed in units of work per unit time.

Examples:

- millions of instructions / sec (MIPS)
- millions of floating point instructions / sec (MFLOPS)
- millions of bytes / sec (MBytes/sec)
- millions of bits / sec (Mbits/sec)
- images / sec
- samples / sec
- transactions / sec (TPS)

Performance expressed as a rate(cont)

Key idea: Report rates that track execution time.

Example: Suppose we are measuring a program that convolves a stream of images from a video camera.

Bad performance measure: MFLOPS

- number of floating point operations depends on the particular convolution algorithm: n^2 matix-vector product vs nlogn fast Fourier transform. An FFT with a bad MFLOPS rate may run faster than a matrix-vector product with a good MFLOPS rate.

Good performance measure: images/sec

- a program that runs faster will convolve more images per second.

Performance expressed as a rate(cont)

Fallacy: Peak rates track running time.

Example: the i860 is advertised as having a peak rate of 80 MFLOPS (40 MHz with 2 flops per cycle).

However, the measured performance of some compiled linear algebra kernels (icc -O2) tells a different story:

Kernel	1d fft	sasum	saxpy	sdot	sgemm	sgemv	spvma
MFLOPS	8.5	3.2	6.1	10.3	6.2	15.0	8.1
%peak	11%	4%	7%	13%	8%	19%	10%

Relating time to system measures

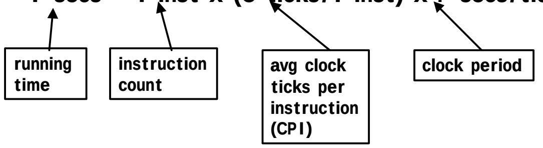
Suppose that for some program we have:

- T seconds running time (the ultimate performance measure)
- C clock ticks, I instructions, P seconds/tick (performance measures of interest to the system designer)

T secs = C ticks x P secs/tick

= (I inst/I inst) x C ticks x P secs/tick

T secs = I inst x (C ticks/I inst) x P secs/tick



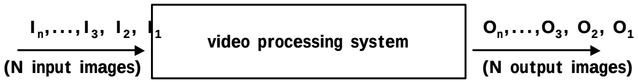
running time

instruction count

avg clock ticks per instruction (CPI)

clock period

Pipeline latency and throughput



$I_n, \ldots, I_3, I_2, I_1$

(N input images)

video processing system

$O_n, \ldots, O_3, O_2, O_1$

(N output images)

Latency (L): time to process an individual image.

Throughput (R): images processed per unit time

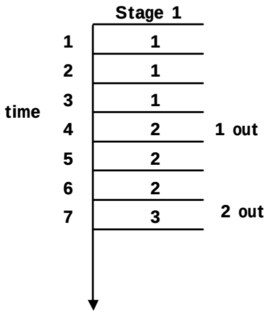
One image can be processed by the system at any point in time

Video system performance

L = 3 secs/image.

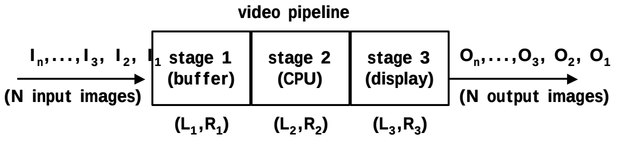
R = 1/L = 1/3 images/sec.

T = L + (N-1)1/R
= 3N



time	Stage 1	
1	1	
2	1	
3	1	
4	2	1 out
5	2	
6	2	
7	3	2 out

Pipelining the video system



One image can be in each stage at any point in time.

L_i = latency of stage i
R_i = throughput of stage i

$L = L_1 + L_2 + L_3$
$R = \min(R_1,\ R_2,\ R_3)$

Pipelined video system performance

Suppose:

$L_1 = L_2 = L_3 = 1$

Then:

L = 3 secs/image.

R = 1 image/sec.

T = L + (N-1)1/R
= N + 2

time	Stage 1	Stage 2	Stage 3	
1	1			
2	2	1		
3	3	2	1	
4	4	3	2	1 out
5	5	4	3	2 out
6	6	5	4	3 out
7	7	6	5	4 out

Relating time to latency & throughput

In general:

- T = L + (N-1)/R

The impact of latency and throughput on running time depends on N:

- (N = 1) => (T = L)
- (N >> 1) => (T = N/R)

To maximize throughput, we should try to maximize the minimum throughput over all stages (i.e., we strive for all stages to have equal throughput).

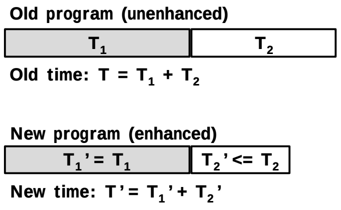
Amdahl's law

You plan to visit a friend in Normandy France and must decide whether it is worth it to take the Concorde SST ($3,100) or a 747 ($1,021) from NY to Paris, assuming it will take 4 hours Pgh to NY and 4 hours Paris to Normandy.

	time NY->Paris	total trip time	speedup over 747
747	8.5 hours	16.5 hours	1
SST	3.75 hours	11.75 hours	1.4

Taking the SST (which is 2.2 times faster) speeds up the overall trip by only a factor of 1.4!

Amdahl’s law (cont)



Old program (unenhanced)

T_1 | T_2

Old time: $T = T_1 + T_2$

New program (enhanced)

$T_1' = T_1$ | $T_2' <= T_2$

New time: $T' = T_1' + T_2'$

T_1 = time that can NOT be enhanced.

T_2 = time that can be enhanced.

T_2' = time after the enhancement.

Speedup: $S_{overall} = T \,/\, T'$

Amdahl’s law (cont)

Two key parameters:

$F_{enhanced} = T_2 \,/\, T$ (fraction of original time that can be improved)

$S_{enhanced} = T_2 \,/\, T_2'$ (speedup of enhanced part)

$$\begin{aligned} T' = T_1' + T_2' = T_1 + T_2' &= T(1-F_{enhanced}) + T_2' \\ &= T(1-F_{enhanced}) + (T_2/S_{enhanced}) && \text{[by def of } S_{enhanced}\text{]} \\ &= T(1-F_{enhanced}) + T(F_{enhanced}\,/S_{enhanced}) && \text{[by def of } F_{enhanced}\text{]} \\ &= T((1-F_{enhanced}) + F_{enhanced}/S_{enhanced}) \end{aligned}$$

Amdahl’s Law:

$S_{overall} = T \,/\, T' = 1/((1-F_{enhanced}) + F_{enhanced}/S_{enhanced})$

Key idea: Amdahl’s law quantifies the general notion of diminishing returns. It applies to any activity, not just computer programs.

Amdahl’s law (cont)

Trip example: Suppose that for the New York to Paris leg, we now consider the possibility of taking a rocket ship (15 minutes) or a handy rip in the fabric of space-time (0 minutes):

	time NY->Paris	total trip time	speedup over 747
747	8.5 hours	16.5 hours	1
SST	3.75 hours	11.75 hours	1.4
rocket	0.25 hours	8.25 hours	2.0
rip	0.0 hours	8 hours	2.1

Amdahl’s law (cont)

Useful corollary to Amdahl’s law:

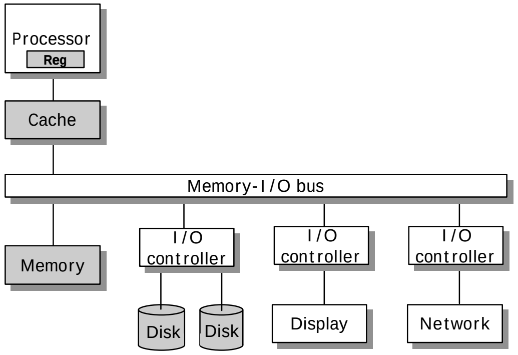
- $1 <= S_{overall} <= 1 \,/\, (1 - F_{enhanced})$

$F_{enhanced}$	Max $S_{overall}$	$F_{enhanced}$	Max $S_{overall}$
0.0	1	0.9375	16
0.5	2	0.96875	32
0.75	4	0.984375	64
0.875	8	0.9921875	128

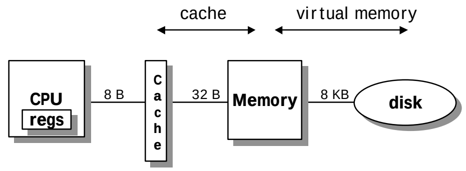
Moral: It is hard to speed up a program.

Moral++ : It is easy to make premature optimizations.

Computer System



Levels in Memory Hierarchy



	Register	Cache	Memory	Disk Memory
size:	200 B	32 KB / 4MB	128 MB	20 GB
speed:	3 ns	6 ns	60 ns	8 ms
$/Mbyte:		$100/MB	$1.50/MB	$0.05/MB
block size:	8 B	32 B	8 KB	

larger, slower, cheaper →

Scaling to 0.1µm

- **Semiconductor Industry Association, 1992 Technology Workshop**
  - Projected future technology based on past trends

	1992	1995	1998	2001	2004	2007
Feature size:	0.5	0.35	0.25	0.18	0.12	0.10
– *Industry is slightly ahead of projection*						
DRAM capacity:	16M	64M	256M	1G	4G	16G
– *Doubles every 1.5 years*						
– *Prediction on track*						
Chip area (cm^2):	2.5	4.0	6.0	8.0	10.0	12.5
– *Way off! Chips staying small*						

Static RAM (SRAM)

Fast

- **~4 nsec access time**

Persistent

- **as long as power is supplied**
- **no refresh required**

Expensive

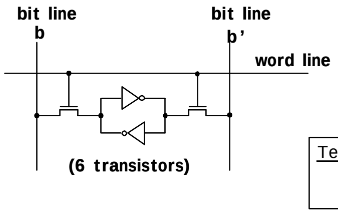
- **~$100/MByte**
- **6 transistors/bit**

Stable

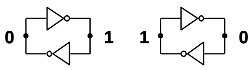
- **High immunity to noise and environmental disturbances**

Technology for caches

Anatomy of an SRAM Cell



Stable Configurations



Terminology:
bit line: carries data
word line: used for addressing

Write:
1. set bit lines to new data value
   - **b'** is set to the opposite of **b**
2. raise word line to "high"

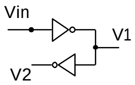
Þ sets cell to new state (may involve flipping relative to old state)

Read:
1. set bit lines high
2. set word line high
3. see which bit line goes low

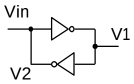
SRAM Cell Principle

Inverter Amplifies
- **Negative gain**
- **Slope < –1 in middle**
- **Saturates at ends**

Inverter Pair Amplifies
- **Positive gain**
- **Slope > 1 in middle**
- **Saturates at ends**



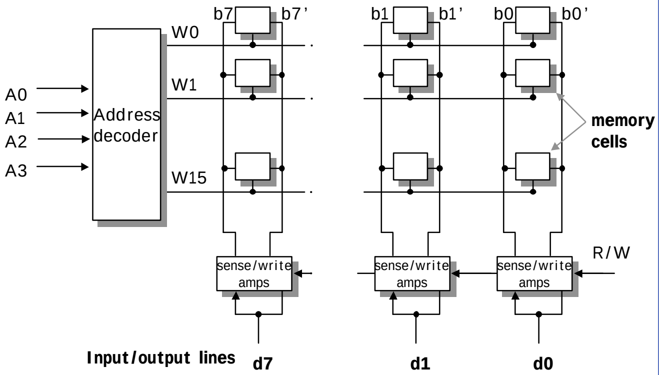
Bistable Element



Stability
- **Require Vin = V2**
- **Stable at endpoints**
  - recover from pertubation
- **Metastable in middle**
  - Fall out when perturbed

Ball on Ramp Analogy

Example SRAM Configuration (16 x 8)



Dynamic RAM (DRAM)

Slower than SRAM

- access time ~60 nsec

Nonpersistant

- every row must be accessed every ~1 ms (refreshed)

Cheaper than SRAM

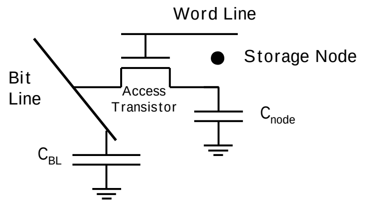
- ~$1.50 / MByte
- 1 transistor/bit

Fragile

- electrical noise, light, radiation

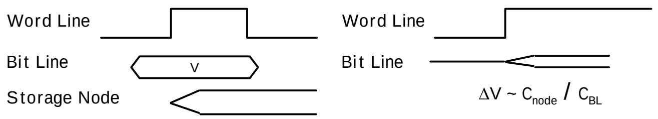
Workhorse memory technology

Anatomy of a DRAM Cell



Writing

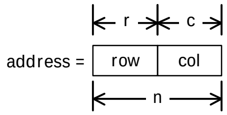
Reading



Addressing Arrays with Bits

Array Size

- R rows, $R = 2^r$
- C columns, $C = 2^c$
- N = R * C bits of memory



Addressing

- Addresses are n bits, where $N = 2^n$
- row(address) = address / C
  - leftmost r bits of address
- col(address) = address % C
  - rightmost bits of address

Example

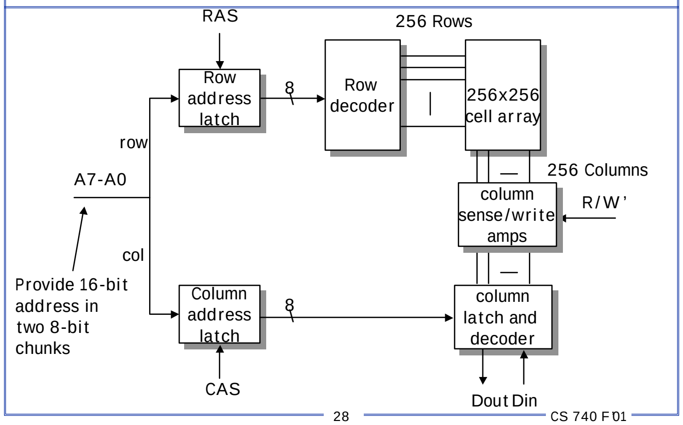
- R = 2
- C = 4
- address = 6

	0	1	2	3
0	000	001	010	011
1	100	101	110	111

row 1

col 2

Example 2-Level Decode DRAM (64Kx1)



DRAM Operation

Row Address (~50ns)

- Set Row address on address lines & strobe RAS
- Entire row read & stored in column latches
- Contents of row of memory cells destroyed

Column Address (~10ns)

- Set Column address on address lines & strobe CAS
- Access selected bit
  - READ: transfer from selected column latch to Dout
  - WRITE: Set selected column latch to Din

Rewrite (~30ns)

- Write back entire row

Observations About DRAMs

Timing

- Access time (= 60ns) < cycle time (= 90ns)
- Need to rewrite row

Must Refresh Periodically

- Perform complete memory cycle for each row
- Approximately once every 1ms
- Sqrt(n) cycles
- Handled in background by memory controller

Inefficient Way to Get a Single Bit

- Effectively read entire row of Sqrt(n) bits

Enhanced Performance DRAMs

Conventional Access

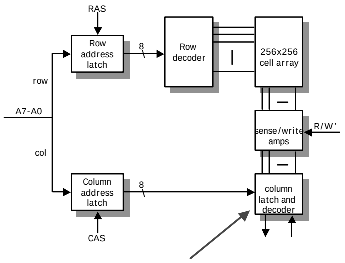
- Row + Col
- RAS CAS RAS CAS ...

Page Mode

- Row + Series of columns
- RAS CAS CAS CAS ...
- Gives successive bits

Other Acronyms

- EDORAM
  - "Extended data output"
- SDRAM
  - "Synchronous DRAM"



Entire row buffered here

Typical Performance

row access time	col access time	cycle time	page mode cycle time
50ns	10ns	90ns	25ns

Video RAM

Performance Enhanced for Video / Graphics Operations

- Frame buffer to hold graphics image

Writing

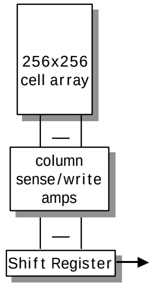
- Random access of bits
- Also supports rectangle fill operations
  - Set all bits in region to 0 or 1

Reading

- Load entire row into shift register
- Shift out at video rates

Performance Example

- 1200 X 1800 pixels / frame
- 24 bits / pixel
- 60 frames / second
- 2.8 GBits / second



Video Stream Output

DRAM Driving Forces

Capacity

- 4X per generation
  - Square array of cells
- Typical scaling
  - Lithography dimensions 0.7X
    - » Areal density 2X
  - Cell function packing 1.5X
  - Chip area 1.33X
- Scaling challenge
  - Typically C_{node} / C_{BL} = 0.1–0.2
  - Must keep C_{node} high as shrink cell size

Retention Time

- Typically 16–256 ms
- Want higher for low-power applications

DRAM Storage Capacitor

Planar Capacitor

- Up to 1Mb
- C decreases linearly with feature size

Trench Capacitor

- 4–256 Mb
- Lining of hole in substrate

Stacked Cell

- > 1Gb
- On top of substrate
- Use high e dielectric

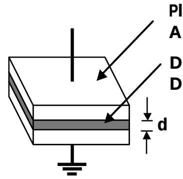


Plate
Area A

Dielectric Material
Dielectric Constant e

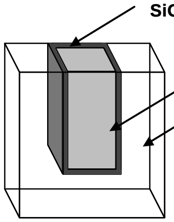
d

C = eA/d

Trench Capacitor

Process

- Etch deep hole in substrate
  - Becomes reference plate
- Grow oxide on walls
  - Dielectric
- Fill with polysilicon plug
  - Tied to storage node



SiO_2 Dielectric

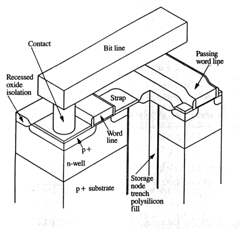
Storage Plate

Reference Plate

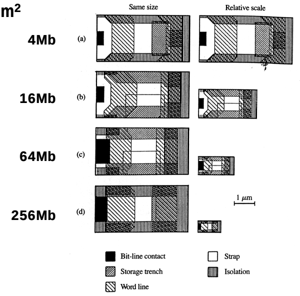
IBM DRAM Evolution

- IBM J. R&D, Jan/Mar '95
- Evolution from 4 – 256 Mb
- 256 Mb uses cell with area 0.6 µm^2

4 Mb Cell Structure



Cell Layouts



Mitsubishi Stacked Cell DRAM

- IEDM 95
- Claim suitable for 1 – 4 Gb

Technology

- 0.14 µm process
  - Synchrotron X-ray source
- 8 nm gate oxide
- 0.29 μm^2 cell

Storage Capacitor

- Fabricated on top of everything else
- Rubidium electrodes
- High dielectric insulator
  - 50X higher than SiO_2
  - 25 nm thick
- Cell capacitance 25 femtofarads

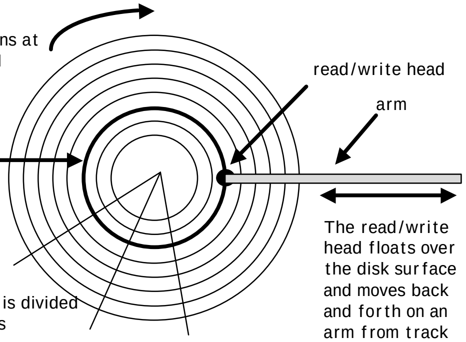
Cross Section of 2 Cells

(Figure removed)

Mitsubishi DRAM Pictures

(Figures removed)

Magnetic Disks



Disk surface spins at 3600–7200 RPM

read/write head

arm

The surface consists of a set of concentric magnetized rings called tracks

Each track is divided into sectors

The read/write head floats over the disk surface and moves back and forth on an arm from track to track.

Disk Capacity

Parameter	18GB Example
• Number Platters	12
• Surfaces / Platter	2
• Number of tracks	6962
• Number sectors / track	213
• Bytes / sector	512
Total Bytes	**18,221,948,928**

Disk Operation

Operation

- Read or write complete sector

Seek

- Position head over proper track
- Typically 6-9ms

Rotational Latency

- Wait until desired sector passes under head
- Worst case: complete rotation

  10,025 RPM Þ 6 ms

Read or Write Bits

- Transfer rate depends on # bits per track and rotational speed
- E.g., 213 * 512 bytes @10,025RPM = 18 MB/sec.
- Modern disks have external transfer rates of up to 80 MB/sec
  - DRAM caches on disk help sustain these higher rates

Disk Performance

Getting First Byte

- Seek + Rotational latency = 7,000 – 19,000 µsec

Getting Successive Bytes

- ~ 0.06 µsec each
  - *roughly 100,000 times faster than getting the first byte!*

Optimizing Performance:

- Large block transfers are more efficient
- Try to do other things while waiting for first byte
  - switch context to other computing task
  - processor is interrupted when transfer completes

Disk / System Interface

1. Processor Signals Controller

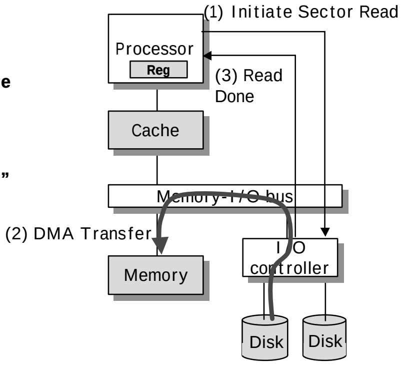
- Read sector X and store starting at memory address Y

2. Read Occurs

- "Direct Memory Access" (DMA) transfer
- Under control of I/O controller

3. I / O Controller Signals Completion

- Interrupts processor
- Can resume suspended process



(1) Initiate Sector Read

Processor

Reg

(3) Read Done

Cache

Memory-I/O bus

(2) DMA Transfer

Memory

I/O controller

Disk

Disk

Magnetic Disk Technology

Seagate ST-12550N Barracuda 2 Disk

Linear density	52,187.	bits per inch (BPI)
– Bit spacing	0.5	microns
Track density	3,047.	tracks per inch (TPI)
– Track spacing	8.3	microns
Total tracks	2,707.	tracks
Rotational Speed	7200.	RPM
Avg Linear Speed	86.4	kilometers / hour
Head Floating Height	0.13	microns

Analogy:

- put the Sears Tower on its side
- fly it around the world, *2.5cm* above the ground
- each complete orbit of the earth takes *8 seconds*

CD Read Only Memory (CDROM)

Basis

- **Optical recording technology developed for audio CDs**
  - 74 minutes playing time
  - 44,100 samples / second
  - 2 X 16-bits / sample (Stereo)
    - Þ Raw bit rate = 172 KB / second
- **Add extra 288 bytes of error correction for every 2048 bytes of data**
  - Cannot tolerate any errors in digital data, whereas OK for audio

Bit Rate

- **172 * 2048 / (288 + 2048) = 150 KB / second**
  - For 1X CDROM
  - *N* X CDROM gives bit rate of *N* * 150
  - E.g., 12X CDROM gives 1.76 MB / second

Capacity

- **74 Minutes * 150 KB / second * 60 seconds / minute = 650 MB**

Storage Trends

SRAM

metric	1980	1985	1990	1995	2000	***2000:1980***
$/MB	19,200	2,900	320	256	100	***190***
access (ns)	300	150	35	15	2	***100***

DRAM

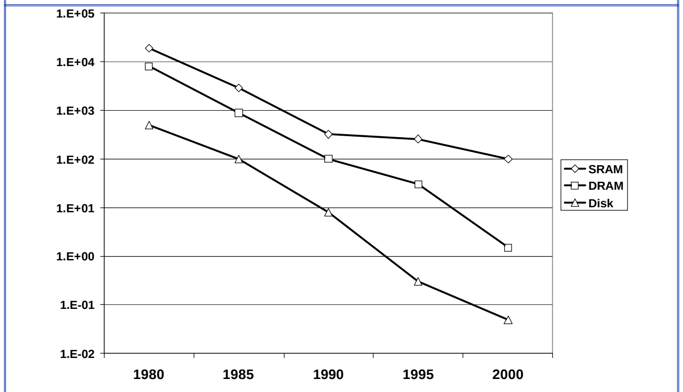
metric	1980	1985	1990	1995	2000	***2000:1980***
$/MB	8,000	880	100	30	1.5	***5,300***
access (ns)	375	200	100	70	60	***6***
typical size(MB)	0.064	0.256	4	16	64	***1,000***

Disk

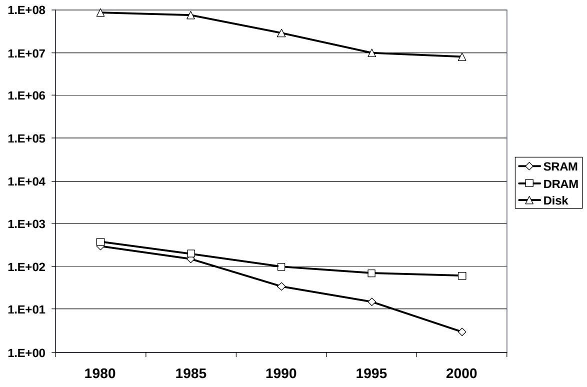
metric	1980	1985	1990	1995	2000	***2000:1980***
$/MB	500	100	8	0.30	0.05	***10,000***
access (ms)	87	75	28	10	8	***11***
typical size(MB)	1	10	160	1,000	9,000	***9,000***

(Culled from back issues of Byte and PC Magazine)

Storage Price: $/MByte



Storage Access Times (nsec)



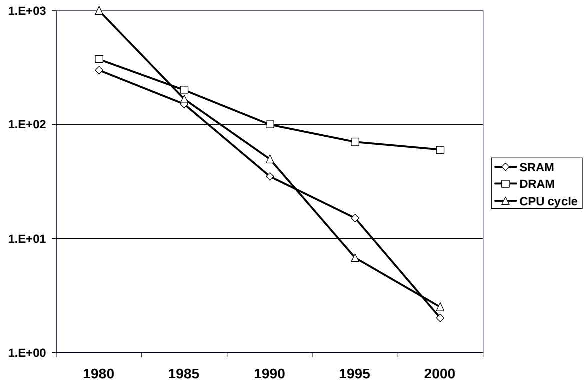
Processor clock rates

Processors

metric	1980	1985	1990	1995	2000	***2000:1980***
typical clock(MHz)	1	6	20	150	600	**600**
processor	8080	286	386	Pentium	P-III	

culled from back issues of Byte and PC Magazine

The CPU vs. DRAM Latency Gap (ns)



Memory Technology Summary

Cost and Density Improving at Enormous Rates

Speed Lagging Processor Performance

Memory Hierarchies Help Narrow the Gap:

- **Small fast SRAMS (cache) at upper levels**
- **Large slow DRAMS (main memory) at lower levels**
- **Incredibly large & slow disks to back it all up**

Locality of Reference Makes It All Work

- **Keep most frequently accessed data in fastest memory**