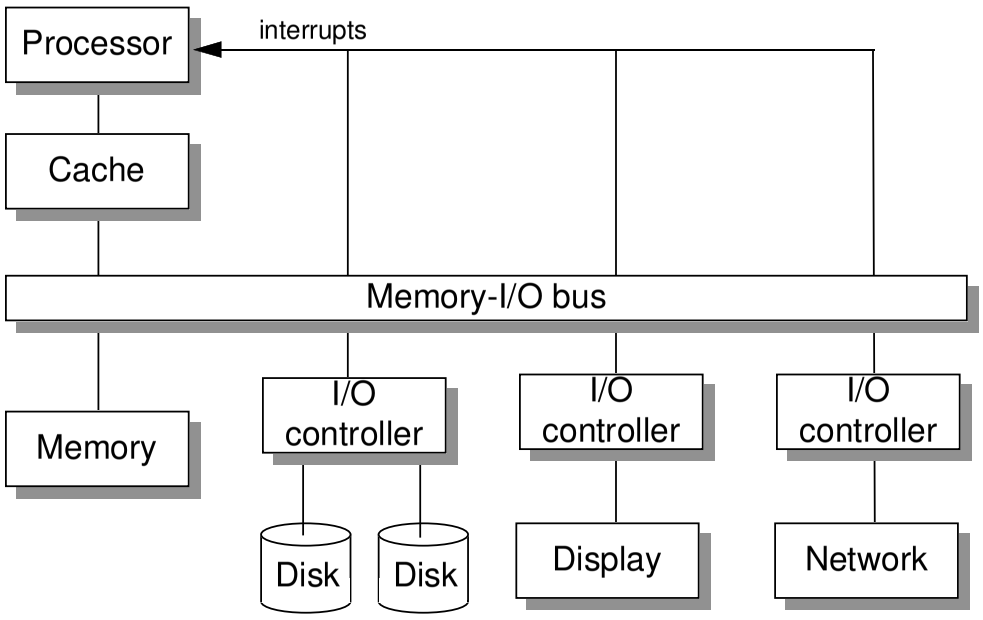
The Memory Hierarchy

April 2, 1996

Topics

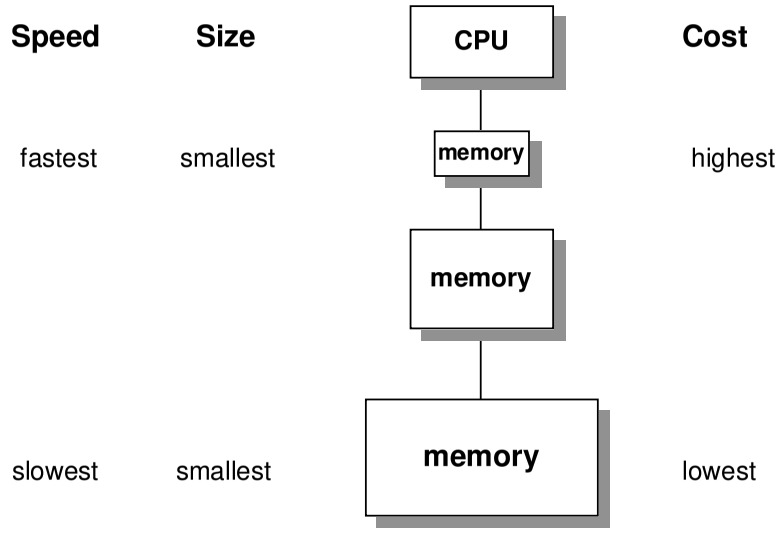
- **memory hierarchy basics**
- **locality of reference**
- **memory technology and trends**

Computer System



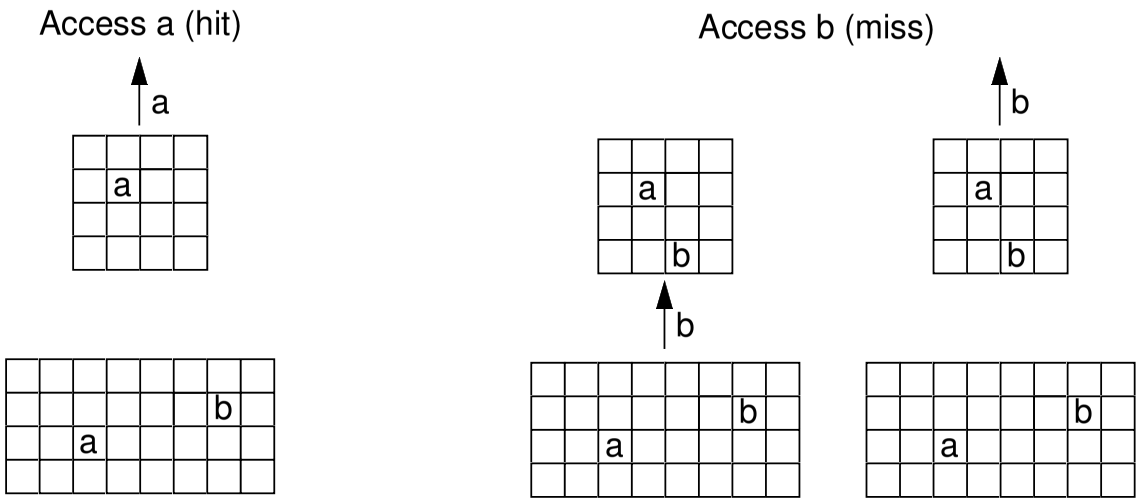
Basic structure of a memory hierarchy

Idea: Exploit locality of reference to give the appearance of fast access to large amounts of memory.

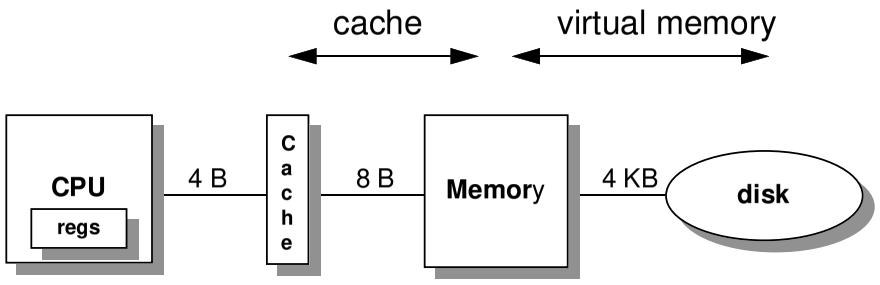


Accessing data in a memory hierarchy

Between any two levels, memory divided into *blocks.*
Data moves between levels on demand, in block-sized chunks.
Upper-level blocks a subset of lower-level blocks.



Levels in a typical memory hierarchy



	register reference	cache reference	memory reference	disk memory reference
size:	200 B	64 KB	32 MB	2 GB
speed:	5 ns	10 ns	100 ns	10 ms
$/Mbyte:		$256/MB	$30/MB	$0.30/MB
block size:	4 B	8 B	4 KB	

larger, slower, cheaper

Basic Forms

Cache

- **all hardware support**
- **handle miss with stall**
- **fast and simple**

Virtual memory (paging)

- **hardware and OS support**
- **handle miss with fault (internal exception)**
- **fundamental basis for many key mechanisms**
  - separate process address spaces
  - relocation
  - protection

Motivation for memory hierarchies

Memory hierarchies are driven by fundamental properties of programs and storage technology:

- **programs:**
  - principle of locality
- **storage technololgy:**
  - price/performance tradeoffs among storage technologies (SRAM, DRAM, and disk)
  - increasing gap between processor and memory performance

Locality of reference

principle of locality: **programs tend to reuse data and instructions they have used recently.**

- ***temporal locality:*** **recently referenced items are likely to be referenced in the near future.**
- ***spatial locality:*** **items with nearby addresses tend to be referenced close together in time.**

good temporal locality
poor spatial locality

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

good temporal and
spatial locality

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

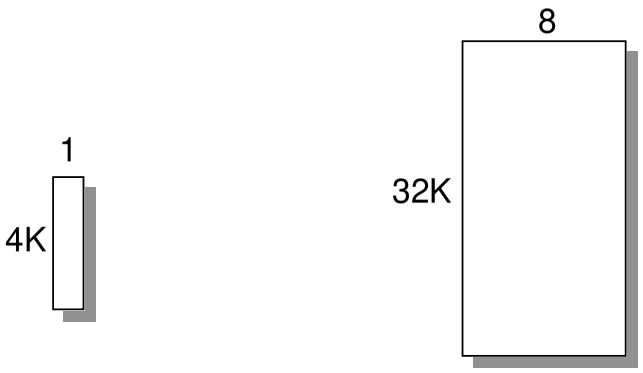
?

```
for (i=0; i<n; i++)
  for (k=i; k<n; k++)
    b[k][i] = a[i][k]
```

Memory components

Memory components characterized by *height x width:*

- **height: number of addressable locations**
- **width: number of bits per location**
- **E.g., 4Kx1, 32Kx8**



Static RAM (SRAM)

Fast

- **~10 ns [1995]**

Persistent

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

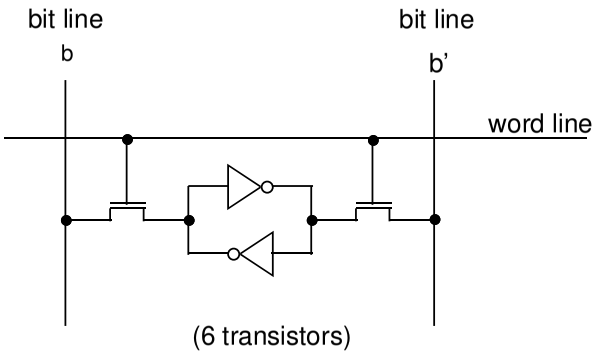
Expensive

- **~$256/MByte [1995]**
- **6 transistors/bit**

Stable

Technology for caches

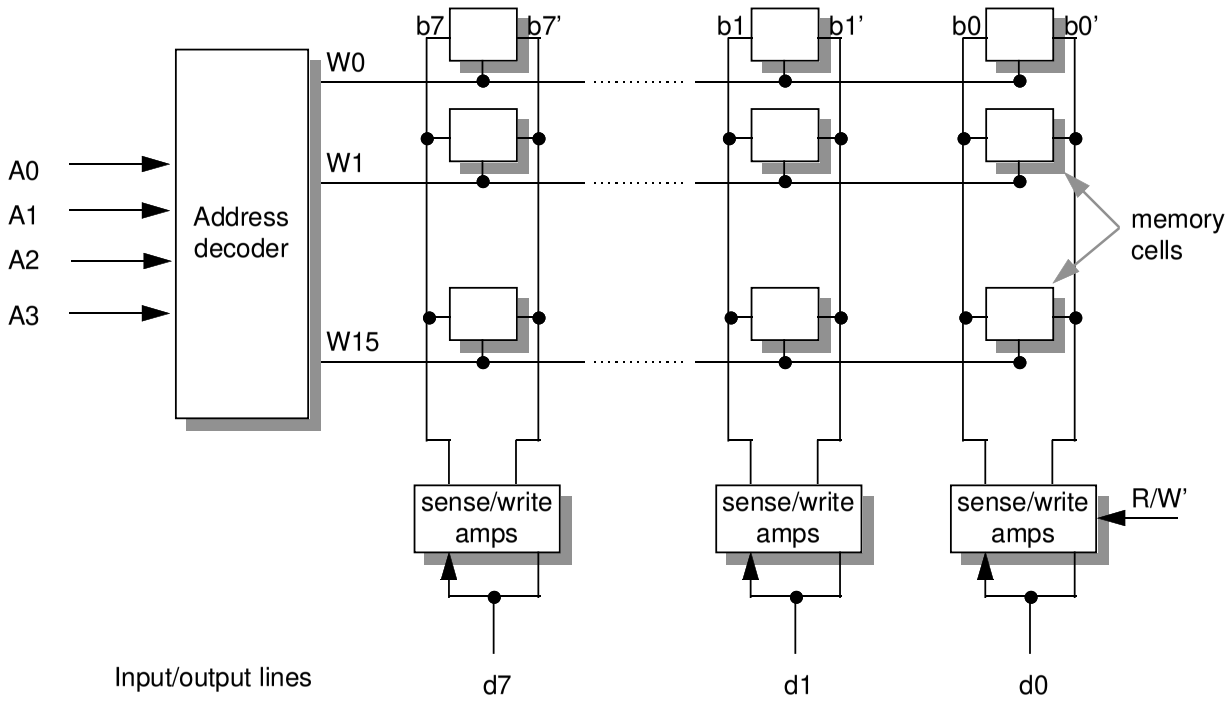
Anatomy of an SRAM bit (cell)



Read:
- set bit lines high
- set word line high
- see which bit line goes low

Write:
- set bit lines to opposite values
- set word line

Example 1-level-decode SRAM (16 x 8)



Dynamic RAM (DRAM)

Slower than SRAM

- access time ~70 ns [1995]

Nonpersistant

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

Cheaper than SRAM

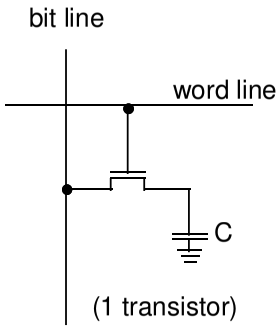
- ~$30/MByte [1995]
- 1 transistor/bit

Fragile

- electrical noise, light, radiation

Workhorse memory technology

Anatomy of a DRAM cell (bit)

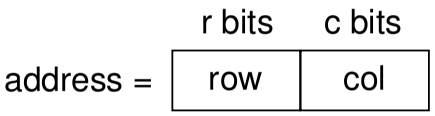


Read:
- set bit line to intermediate value(30 fF)
- set word line high
- sense blip (130 mV)
- set result
- set bit line (refresh)

Write:
- set bit line
- set word line

Addessing arrays with bits

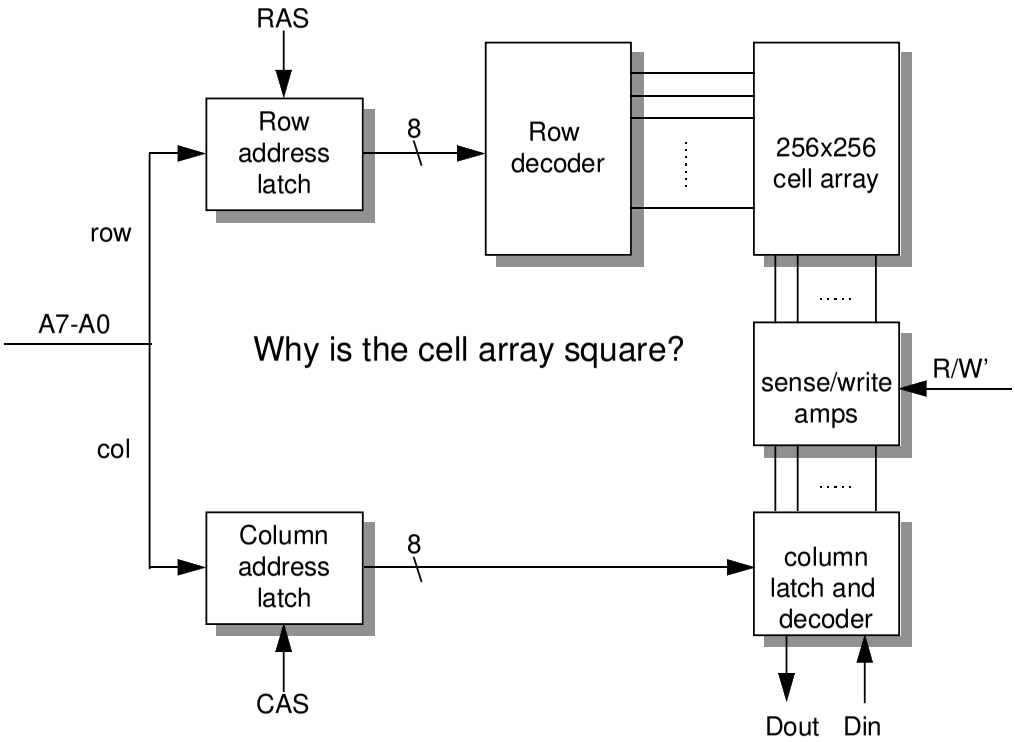
Consider an R x C array of addresses, where R = 2^r and C = 2^c.
Then for each address,
row(address) = address / C = leftmost r bits of address
col(address) = address % C = righmost c bits of address



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

row 1 col 2

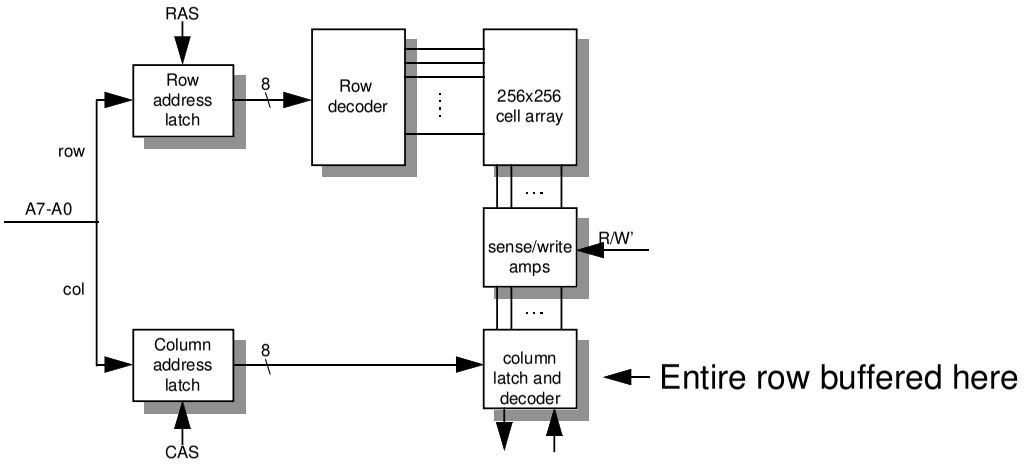
Example 2-level decode DRAM (64Kx1)



Page mode DRAM (fast block transfers)

For words in same row, instead of:
RAS CAS RAS CAS ...

Send the row address once, followed by arbitrary column accesses until the start of the next refresh cycle:
RAS CAS CAS CAS ...



DRAM access time vs cycle time

access time: time between when a word is requested and when it is available.
cycle time: minimum time between requests to memory.

For DRAMs, access time < cycle time because of refresh.
Typically, for n bit DRAM, refresh requires sqrt(n) accesses/1 ms.

Page mode accesses can greatly reduce the cycle time.
Example (16Mx1 DRAM):

row access time	col access time	cycle time	page mode cycle time
65ns	15ns	140ns	30ns

Locality of reference strikes again!

Storage trends (memory)

SRAM

metric	1980	1985	1990	1995	1995:1980
$/MB	19,200	2,900	320	256	75
access (ns)	300	150	35	15	20

DRAM

metric	1980	1985	1990	1995	1995:1980
$/MB	8,000	880	100	30	266
access (ns)	375	200	100	70	5
typical size(MB)	0.064	0.256	4	16	250

culled from back issues of Byte and PC Magazine

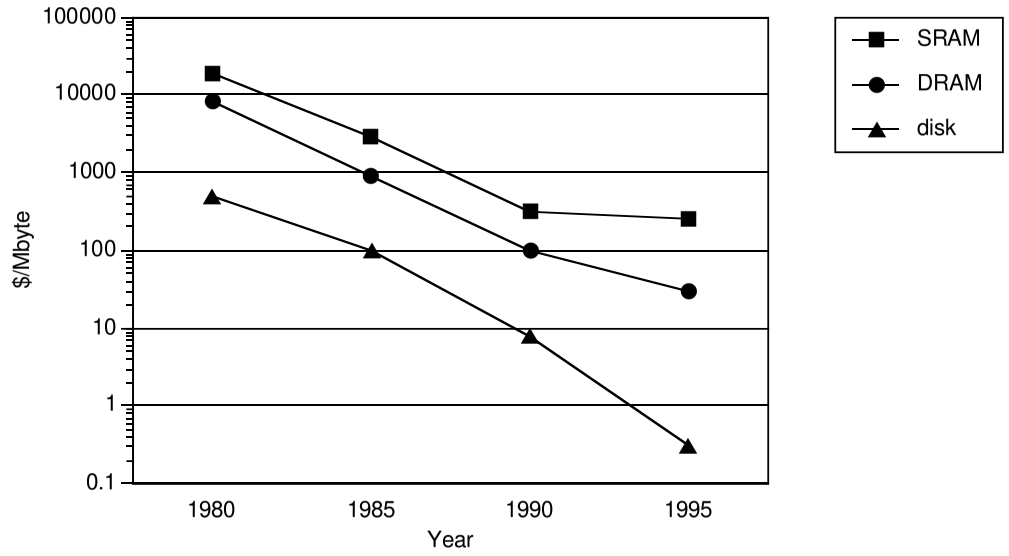
Storage trends (disk)

Disks

metric	1980	1985	1990	1995	1995:1980
$/MB	500	100	8	0.30	1,600
access (ms)	87	75	28	10	9
typical size(MB)	1	10	160	1,000	1,000

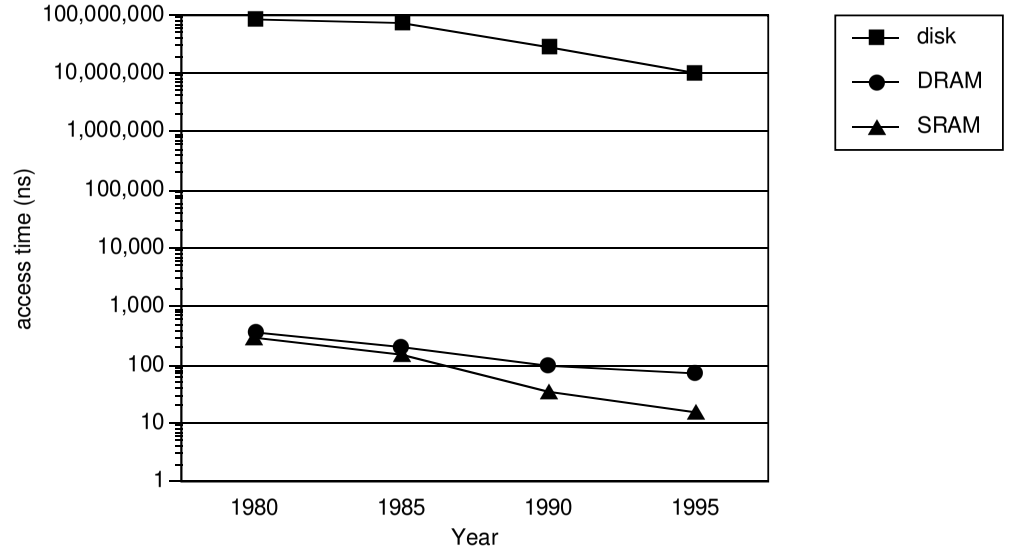
culled from back issues of Byte and PC Magazine

Storage price/MByte



culled from back issues of Byte and PC Magazine

Storage access times



culled from back issues of Byte and PC Magazine

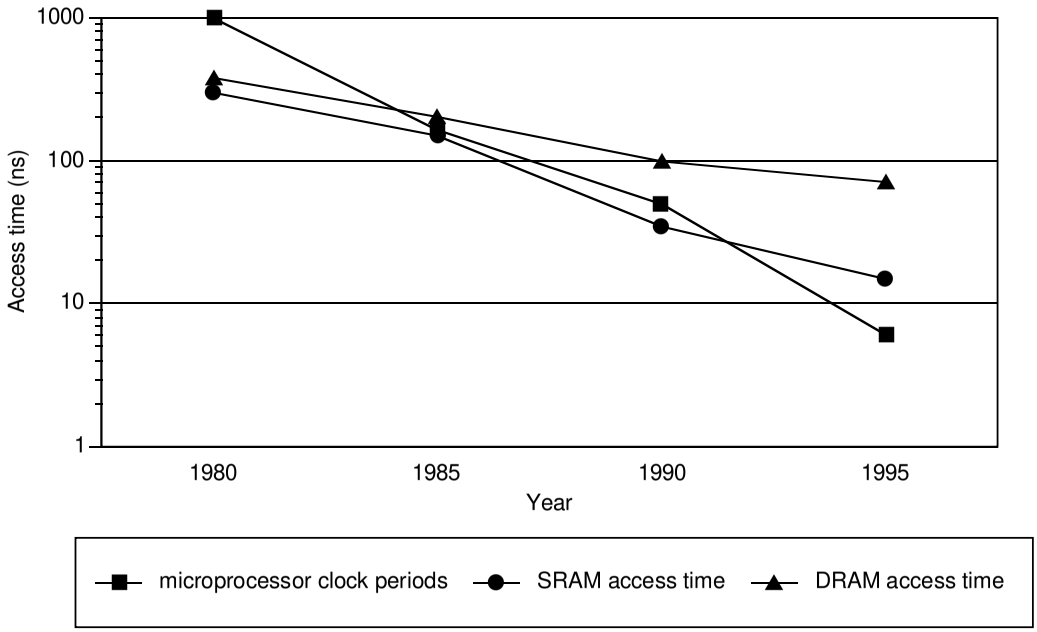
Processor clock rates

Processors

metric	1980	1985	1990	1995	1995:1980
typical clock(MHz)	1	6	20	150	150
processor	8080	286	386	pentium	

culled from back issues of Byte and PC Magazine

The widening processor/memory gap



culled from back issues of Byte and PC Magazine

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**

Locality of reference makes it all work.