332
Advanced Computer Architecture
Chapter 2

Review of Virtual Memory, Cost, Integrated Circuits, Benchmarks, Moore's Law

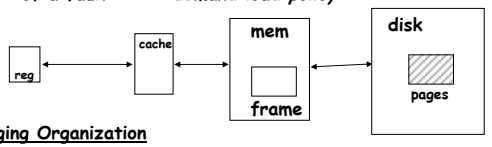
January 2004

Paul H J Kelly

These lecture notes are partly based on the course text, Hennessy and Patterson's *Computer Architecture, a quantitative approach (3rd ed)*, and on the lecture slides of David Patterson's Berkeley course (CS252, Jan 2001)

Basic Issues in VM System Design

- size of information blocks that are transferred from secondary to main storage (M)
- block of information brought into M, and M is full, then some region of M must be released to make room for the new block --> *replacement policy*
- which region of M is to hold the new block --> *placement policy*
- missing item fetched from secondary memory only on the occurrence of a fault --> *demand load policy*



Paging Organization

virtual and physical address space partitioned into blocks of equal size

page frames

pages

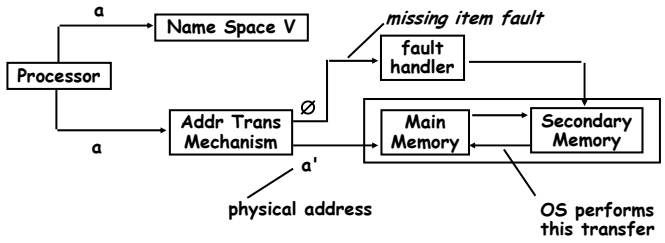
Address Map

V = {0, 1, . . . , n – 1} virtual address space
M = {0, 1, . . . , m – 1} physical address space n > m

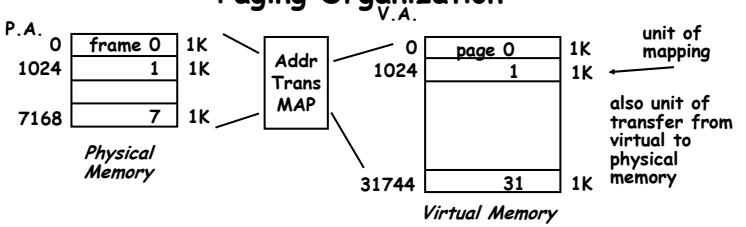
MAP: V --> M U {∅} address mapping function

MAP(a) = a' if data at virtual address a is present in physical address a' and a' is present in M

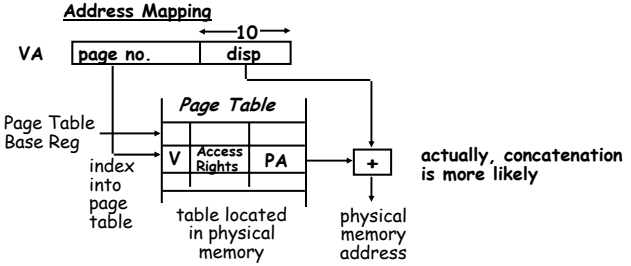
= ∅ if data at virtual address a is not present in M



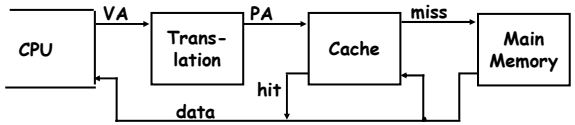
Paging Organization



Address Mapping



Address translation before cache access

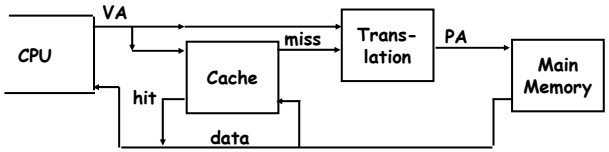


It takes an extra memory access to translate VA to PA

This makes cache access very expensive, and this is the "innermost loop" that you want to go as fast as possible

So it's not a feasible option…

Cache access before address translation

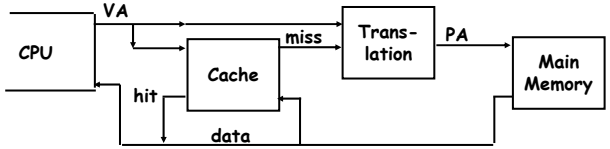


Why access cache with PA at all?

Access cache with VA, translate only on cache miss

Unfortunately there are problems….

Cache access before address translation: aliasing problems



Problems with accessing cache using virtual addresses:

Synonym aliasing problem: two different virtual addresses map to same physical address => two different cache entries holding data for the same physical address!

For update: must update all cache entries with same physical address or memory becomes inconsistent

Determining this requires significant hardware: essentially an associative lookup on the physical address tags to see if you have multiple hits

Homonym aliasing problem: after a context-switch from process A to process B, the VA to PA mapping changes. A VA address x in process A may refer to a different value in process B.

Can these problems be avoided in software, by the operating system?

TLBs

A way to speed up translation is to use a special cache of recently used page table entries -- this has many names, but the most frequently used is *Translation Lookaside Buffer* or *TLB*

Virtual Address	Physical Address	Dirty	Ref	Valid	Access

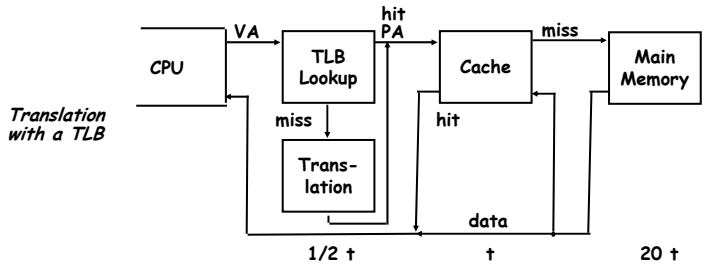
Really just a cache on the page table mappings

TLB access time comparable to cache access time (much less than main memory access time)

Translation Look-Aside Buffers

Just like any other cache, the TLB can be organized as fully associative, set associative, or direct mapped

TLBs are usually small, typically not more than 128 - 256 entries even on high end machines. This permits fully associative lookup on these machines. Most mid-range machines use small n-way set associative organizations.



Translation with a TLB

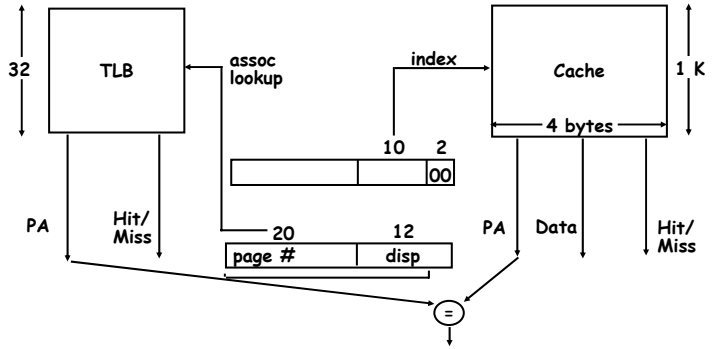
Reducing Translation Time

Machines with TLBs go one step further to reduce # cycles/cache access

They overlap the cache access with the TLB access:

high order bits of the VA are used to look in the TLB while low order bits are used as index into cache

Overlapped Cache & TLB Access



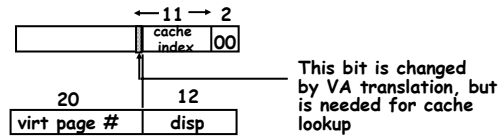
IF cache hit AND (cache tag = PA) then deliver data to CPU
ELSE IF [cache miss OR (cache tag = PA)] and TLB hit THEN
access memory with the PA from the TLB
ELSE do standard VA translation

Problems With Overlapped TLB Access

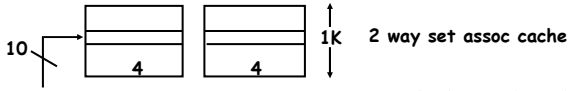
Overlapped access only works as long as the address bits used to index into the cache *do not change* as the result of VA translation

This usually limits things to small caches, large page sizes, or high n-way set associative caches if you want a large cache

Example: suppose everything the same except that the cache is increased to 8 K bytes instead of 4 K:



Solutions:
go to 8K byte page sizes;
go to 2 way set associative cache; or
SW guarantee VA[13]=PA[13]



SPEC: System Performance Evaluation Cooperative

- First Round 1989
  - 10 programs yielding a single number ("SPECmarks")
- Second Round 1992
  - SPECInt92 (6 integer programs) and SPECfp92 (14 floating point programs)
    - Compiler Flags unlimited. March 93 of DEC 4000 Model 610:

      `spice:` `unix.c:/def=(sysv,has_bcopy,"bcopy(a,b,c)= memcpy(b,a,c)"`

      `wave5:` `/ali=(all,dcom=nat)/ag=a/ur=4/ur=200`

      `nasa7:` `/norecu/ag=a/ur=4/ur2=200/lc=blas`
- Third Round 1995
  - new set of programs: SPECint95 (8 integer programs) and SPECfp95 (10 floating point)
  - "benchmarks useful for 3 years"
  - Single flag setting for all programs: SPECint_base95, SPECfp_base95

SPEC: System Performance Evaluation Cooperative

- Fourth Round 2000: SPEC CPU2000
  - 12 Integer
  - 14 Floating Point
  - 2 choices on compilation;
    - "aggressive" (SPECint2000,SPECfp2000),
    - "conservative" (SPECint_base2000,SPECfp_base);

      flags same for all programs, no more than 4 flags, same compiler for conservative, can change for aggressive
  - multiple data sets so that can train compiler if trying to collect data for input to compiler to improve optimization
- Homework: check out the SPEC website:

  www.specbench.org

How to Summarize Performance

- Arithmetic mean (weighted arithmetic mean) tracks execution time:

  $\Sigma(T_i)/n$ or $\Sigma(W_i*T_i)$
- Harmonic mean (weighted harmonic mean) of rates (e.g., MFLOPS) tracks execution time:

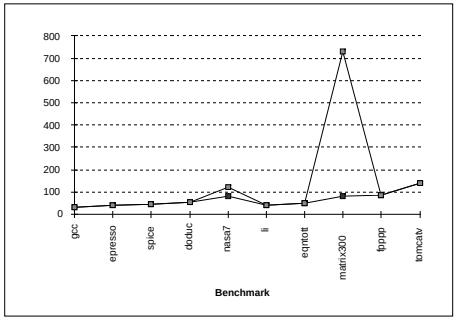
  $n/\Sigma(1/R_i)$ or $n/\Sigma(W_i/R_i)$
- Normalized execution time is handy for scaling performance (e.g., X times faster than SPARCstation 10)
- But do not take the arithmetic mean of normalized execution time, use the geometric mean:

  $(\Pi T_j / N_j)^{1/n}$

(H&P 3rd ed pp.36 is essential reading… "This section should be read by anyone who has **uttered** the words "**performance**" and "**computer**" in the same breath" – Robert Bernecky, book review, Doctor Dobb's Journal)

SPEC First Round

- One program: 99% of time in single line of code
- New front-end compiler could improve dramatically



Impact of Means on SPECmark89 for IBM 550

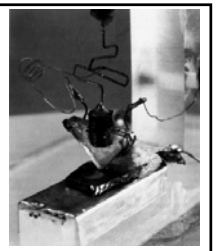
	Ratio to VAX:		Time:		Weighted Time:	
Program	*Before*	*After*	*Before*	*After*	*Before*	*After*
gcc	30	29	49	51	8.91	9.22
espresso	35	34	65	67	7.64	7.86
spice	47	47	510	510	5.69	5.69
doduc	46	49	41	38	5.81	5.45
nasa7	78	144	258	140	3.43	1.86
li	34	34	183	183	7.86	7.86
eqntott	40	40	28	28	6.68	6.68
matrix300	78	730	58	6	3.43	0.37
fpppp	90	87	34	35	2.97	3.07
tomcatv	33	138	20	19	2.01	1.94
Mean	54	72	124	108	54.42	49.99
	Geometric		*Arithmetic*		*Weighted Arith.*	
	Ratio	1.33	Ratio	1.16	Ratio	1.09

Performance Evaluation

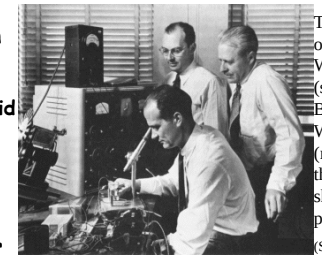
- "For better or worse, benchmarks shape a field"
- Good products created when have:
  - Good benchmarks
  - Good ways to summarize performance
- Given sales is a function in part of performance relative to competition, investment in improving product as reported by performance summary
- If benchmarks/summary inadequate, then choose between improving product for real programs vs. improving product to get more sales; Sales almost always wins!
- Execution time is the measure of computer performance!

A little history…early days at Bell Labs

- 1940: Russell Ohl develops PN junction (accidentally…)
- 1945: Shockley's lab established
- 1947: Bardeen and Brattain create point-contact transistor with two PN junctions, gain=18
- 1951: Shockley develops junction transistor which can be manufactured in quantity
- 1952: British radar expert GWA Dummer forecasts "solid block [with] layers of insulating, conducting and amplifying materials
- 1954: first transistor radio. Also Texas Instruments makes first silicon transistor (price $2.50)



First point-contact transistor invented at Bell Labs. (Source: Bell Labs.)

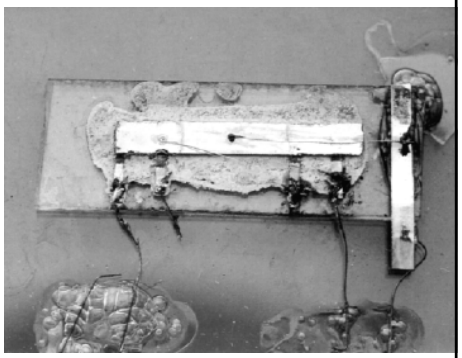


The three inventors of the transistor: William Shockley, (seated), John Bardeen (left) and Walter Brattain (right) in 1948; the three inventors shared the Nobel prize in 1956. (Source: Bell Labs.)

Source: http://6371.lcs.mit.edu/Fall96/lectures/L1/P005.html ; See also http://www.maxmon.com/1952ad.htm

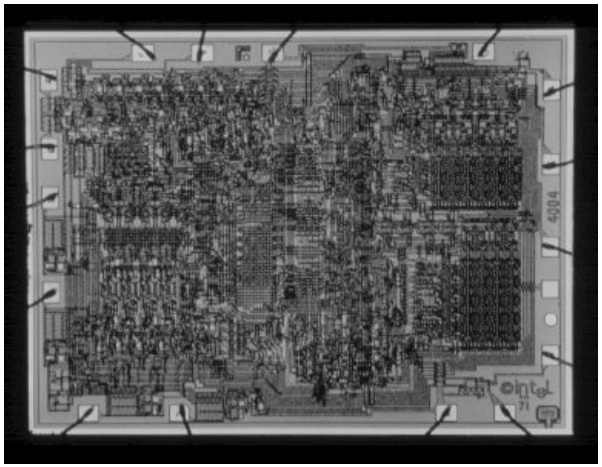
Pre-historic integrated circuits

- 1958: The first monolithic integrated circuit, about the size of a finger tip, developed at Texas Instruments by Jack Kilby. The IC was a chip of a single Germanium crystal containing one transistor, one capacitor, and one resistor (Source: Texas Instruments)

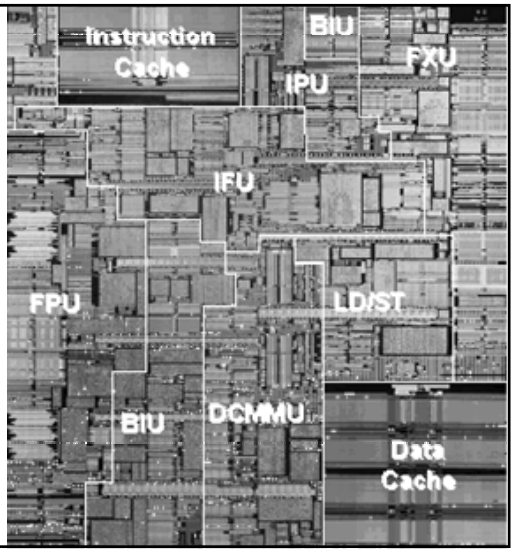


Source: http://kasap3.usask.ca/server/kasap/photo1.html

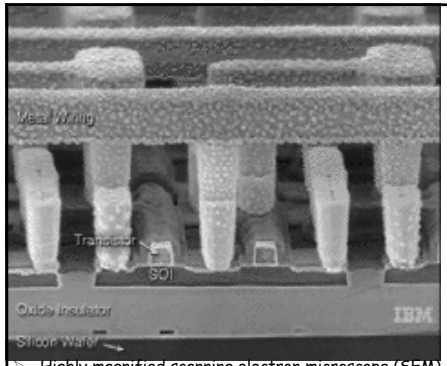
- 1970: Intel starts selling a 1K bit RAM
- 1971: Intel introduces first microprocessor, the 4004
  - 4-bit buses
  - Clock rate 108 KHz
  - 2300 transistors
  - 10μm process

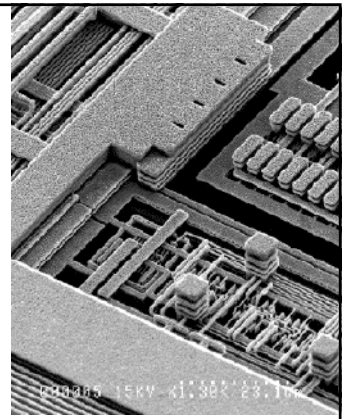


- IBM Power3 microprocessor
- 15M transistors
- 0.18μm copper/SOI process
- About 270mm^2



- Chips are made from slices of a single-crystal silicon ingot
- Each slice is about 30cm in diameter, and 250-600 microns thick
- Transistors and wiring are constructed by photolithography
- Essentially a printing/etching process
- With lines ca. 0.18μm wide



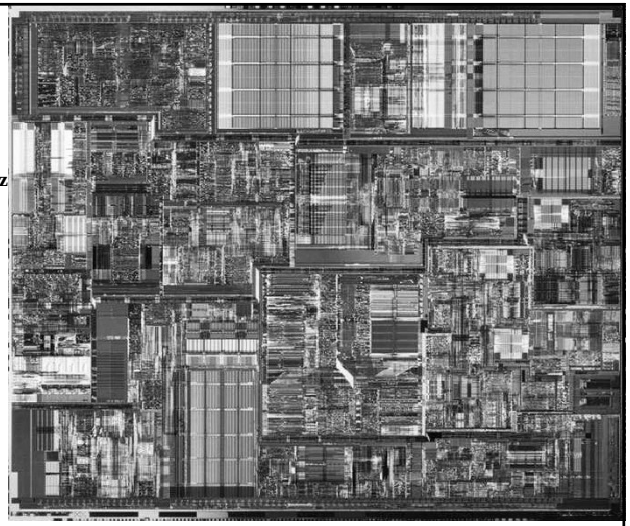


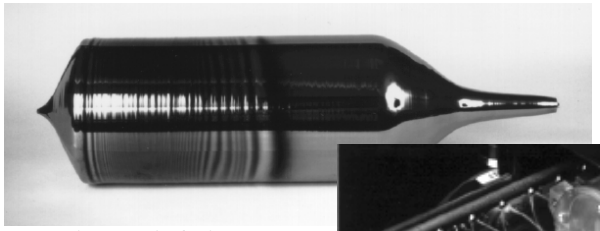
➢ Highly magnified scanning electron microscope (SEM) view of IBM's Silicon-On-Insulator (SOI) fabrication: a layer of silicon crystal is grown on top of a layer of insulating silicon oxide

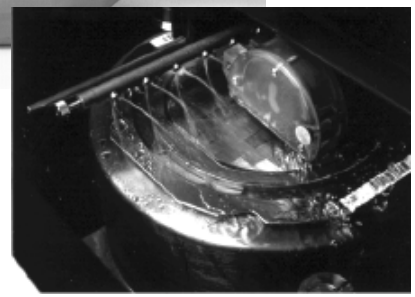
- **Highly magnified scanning electron microscope (SEM) view of IBM's six-level copper interconnect technology in an integrated circuit chip. The aluminum in transistor interconnections in a silicon chip has been replaced by copper that has a higher conductivity (by nearly 40%) and also a better ability to carry higher current densities without electromigration. Lower copper interconnect resistance means higher speeds and lower RC constants (Photograph courtesy of IBM Corporation, 1997.)**

Intel Pentium 4

- **42 M transistors**
- **0.13mm copper/SOI process**
- **Clock speeds: 2200, 2000MHz**
- **Die size 146 square mm**
- **Power consumption 55.1W (2200), 52.4W (2000)**
- **Price ($ per chip, in 1,000-chip units, Jan 2002):**
  - **US$562 (2200)**
  - **US$364 (2000)**





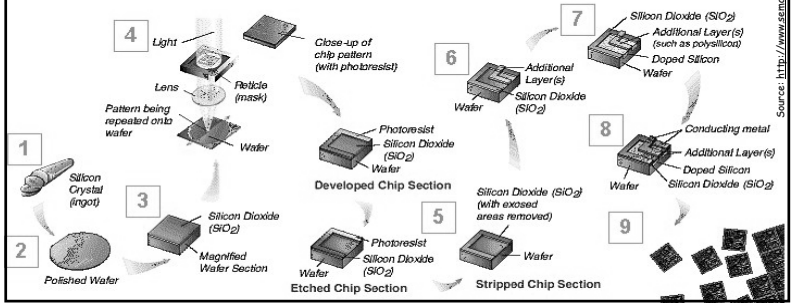


- **A single crystal of silicon, a silicon ingot, grown by the Czochralski technique. The diameter of this ingot is 6 inches (Courtesy of Texas Instruments). State of the art fabs now use 300mm wafers**

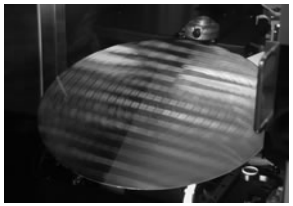
Wafer saw: Each wafer is cut into many individual die using a diamond-edge saw with a cutting edge about the thickness of a human hair. (Photograph courtesy of Micron Technology, Inc., Boise, Idaho)

Integrated circuit fabrication is a printing process

1. Grow pure silicon crystal
2. Slice into wafers and polish
3. Grow surface layer of silicon dioxide (ie glass), either using high-temperature oxygen or chemical vapour deposition
4. Coat surface with photoresist layer, then use mask to selectively expose photoresist to ultraviolet light
5. Etch away silicon dioxide regions not covered by hardened photoresist
6. Further photolithography steps build up additional layers, such as polysilicon
7. Exposed silicon is doped with small quantities of chemicals which alter its semiconductor behaviour to create transistors
8. Further photolithography steps build layers of metal for wiring
9. Die are tested, diced, tested and packaged



Source: http://www.siemtech.org/public/news/mfgprocs/mfgprocshtm



Close up of the wafer as it spins during a testing procedure

Checking wafers processing in a vertical diffusion furnace

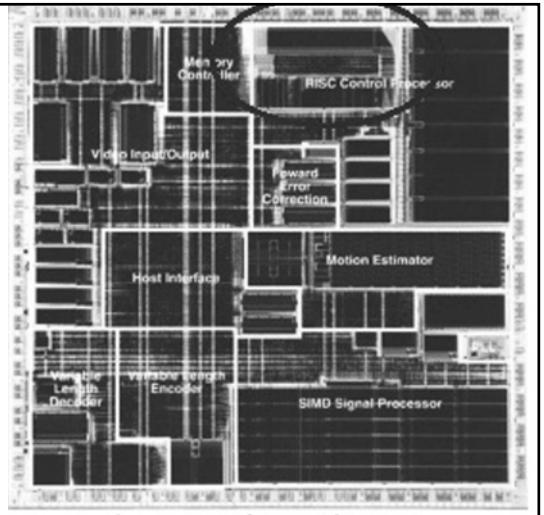
Intel technicians monitor wafers in an automated wet etch tool. The process cleans the wafers of any excess process chemicals or contamination.o)

Micron Technology, Inc., Boise, Idaho)

Source: http://www.intel.com/pressroom/archive/photos/manufacturing_photos.htm

The future

- More transistors
- Higher clock rates
- Lower power
- System-on-a-chip
- Field-programmable gate arrays
- "Compiling to silicon"
- Optical interconnect
- Quantum computing?



AVP-III Video Codec from Lucent Technologies

Source: http://6371.lcs.mit.edu/Fall96/lectures/L1/P005.html

Intel x86/Pentium Family

CPU	Year	Data Bus	Max. Mem.	Transistors	Clock MHz	Av. MIPS	Level-1 Caches
8086	1978	16	1MB	29K	5-10	0.8	
80286	1982	16	16MB	134K	8-12	2.7	
80386	1985	32	4GB	275K	16-33	6	
80486	1989	32	4GB	1.2M	25-100	20	8Kb
Pentium	1993	64	4GB	3.1M	60-233	100	8K Instr + 8K Data
Pentium Pro	1995	64	64GB	5.5M +15.5M	150-200	440	8K + 8K , Level2
Pentium II	1997	64	64GB	7M	266-450	466-	16K+16K + L2
Pentium III	1999	64	64GB	8.2M	500-1000	1000-	16K+16K + L2
Pentium IV	2001	64	64GB	42M	1300-2000		8K + L2

On-line manuals: http://x86.ddj.com/intel.doc/386manuals.htm

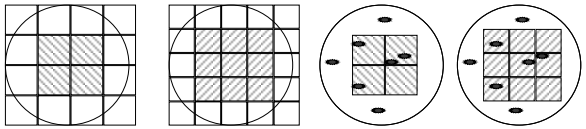
On-line details: http://www.sandpile.org/ia32/index.htm

Integrated Circuits Costs

$$\text{IC cost} = \frac{\text{Die cost} + \text{Testing cost} + \text{Packaging cost}}{\text{Final test yield}}$$

$$\text{Die cost} = \frac{\text{Wafer cost}}{\text{Dies per Wafer} \times \text{Die yield}}$$

$$\text{Dies per wafer} = \frac{\pi\,(\text{Wafer\_dia}\ m/2)^2}{\text{Die\_Area}} - \frac{\pi \times \text{Wafer\_diam}}{\sqrt{2 \cdot \text{Die\_Area}}} - \text{Test\_Die}$$



$$\text{Die Yield} = \text{Wafer\_yield} \times \left\{1 - \left(\frac{\text{Defect\_Density} \times \text{Die\_area}}{\alpha}\right)^{-\alpha}\right\}$$

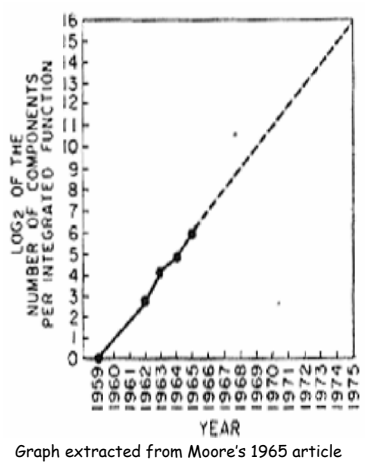
Die Cost goes roughly with die area^4

Real World Examples

Chip	Metal layers	Line width	Wafer cost	Defect /cm^2	Area mm^2	Dies/ wafer	Yield	Die Cost
386DX	2	0.90	$900	1.0	43	360	71%	$4
486DX2	3	0.80	$1200	1.0	81	181	54%	$12
PowerPC 601	4	0.80	$1700	1.3	121	115	28%	$53
HP PA 7100	3	0.80	$1300	1.0	196	66	27%	$73
DEC Alpha	3	0.70	$1500	1.2	234	53	19%	$149
SuperSPARC	3	0.70	$1700	1.6	256	48	13%	$272
Pentium	3	0.80	$1500	1.5	296	40	9%	$417

From "Estimating IC Manufacturing Costs," by Linley Gwennap, *Microprocessor Report*, August 2, 1993, p. 15

Moore's "Law"



Graph extracted from Moore's 1965 article

Cramming more components onto integrated circuits
By Gordon E. Moore

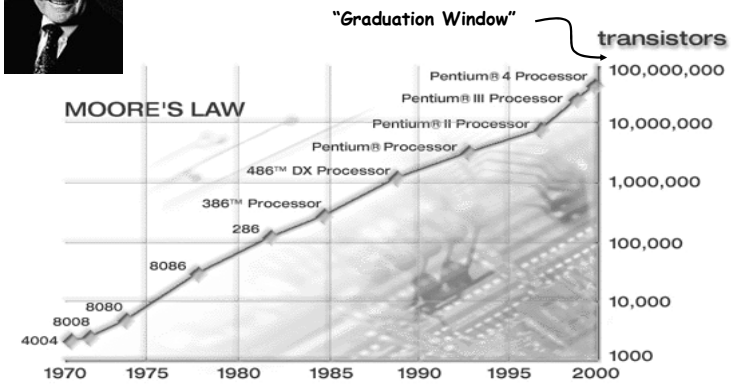
Electronics, Volume 38, Number 8, April 19, 1965
(See http://www.intel.com/research/silicon/moorslaw.htm)

"With unit cost falling as the number of components per circuit rises, by 1975 economics may dictate squeezing as many as 65,000 components on a single silicon chip"

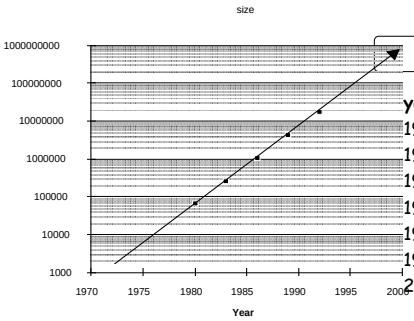
Gordon Moore left Fairchild to found Intel in 1968 with Robert Noyce and Andy Grove,

Technology Trends: Microprocessor Capacity

CMOS improvements:
- **Die size: 2X every 3 yrs**
- **Line width: halve / 7 yrs**



Memory Capacity (Single Chip DRAM)

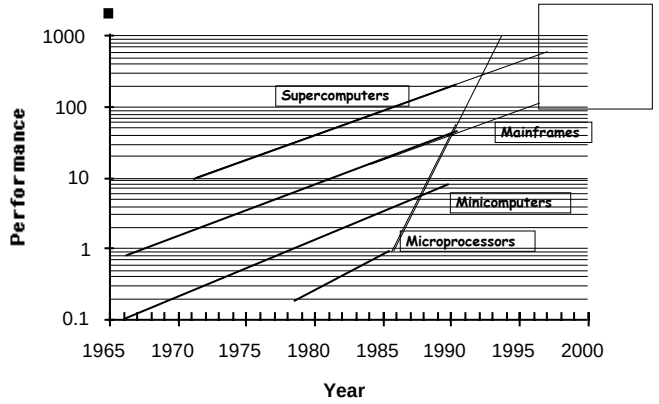


year	size(Mb)	cyc time
1980	0.0625	250 ns
1983	0.25	220 ns
1986	1	190 ns
1989	4	165 ns
1992	16	145 ns
1996	64	120 ns
2000	256	100 ns

Technology Trends (Summary)

	Capacity	Speed (latency)
Logic	2x in 3 years	2x in 3 years
DRAM	4x in 3-4 years	2x in 10 years
Disk	4x in 2-3 years	2x in 10 years

Processor Performance Trends



Review #3/3: TLB, Virtual Memory

- Caches, TLBs, Virtual Memory all understood by examining how they deal with 4 questions:
  1. Where can block be placed?
  2. How is block found?
  3. What block is replaced on miss?
  4. How are writes handled?
- Page tables map virtual address to physical address
- TLBs make virtual memory practical
  - Locality in data => locality in addresses of data, temporal and spatial
- TLB misses are significant in processor performance
  - funny times, as most systems can't access all of 2nd level cache without TLB misses!
- Today VM allows many processes to share single memory without having to swap all processes to disk; today VM protection is more important than memory hierarchy

Summary

- Performance Summary needs good benchmarks and good ways to summarize performance
- Transistors/chip for microprocessors growing via "Moore's Law" 2X 1.5/yrs
- Disk capacity (so far) is growing at a faster rate over the last 4-5 years
- DRAM capacity is growing at a slower rate over the last 4-5 years
- In general, Bandwidth improving fast, latency improving slowly