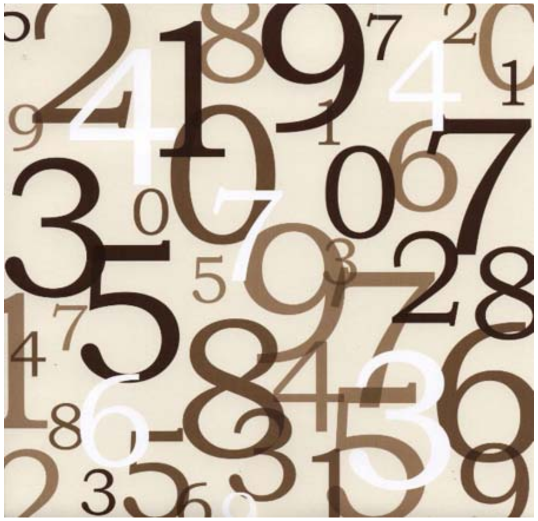
inst.eecs.berkeley.edu/~cs61c

CS61C : Machine Structures

Lecture #1 – Introduction & Numbers



2006-06-26

Andy Carle

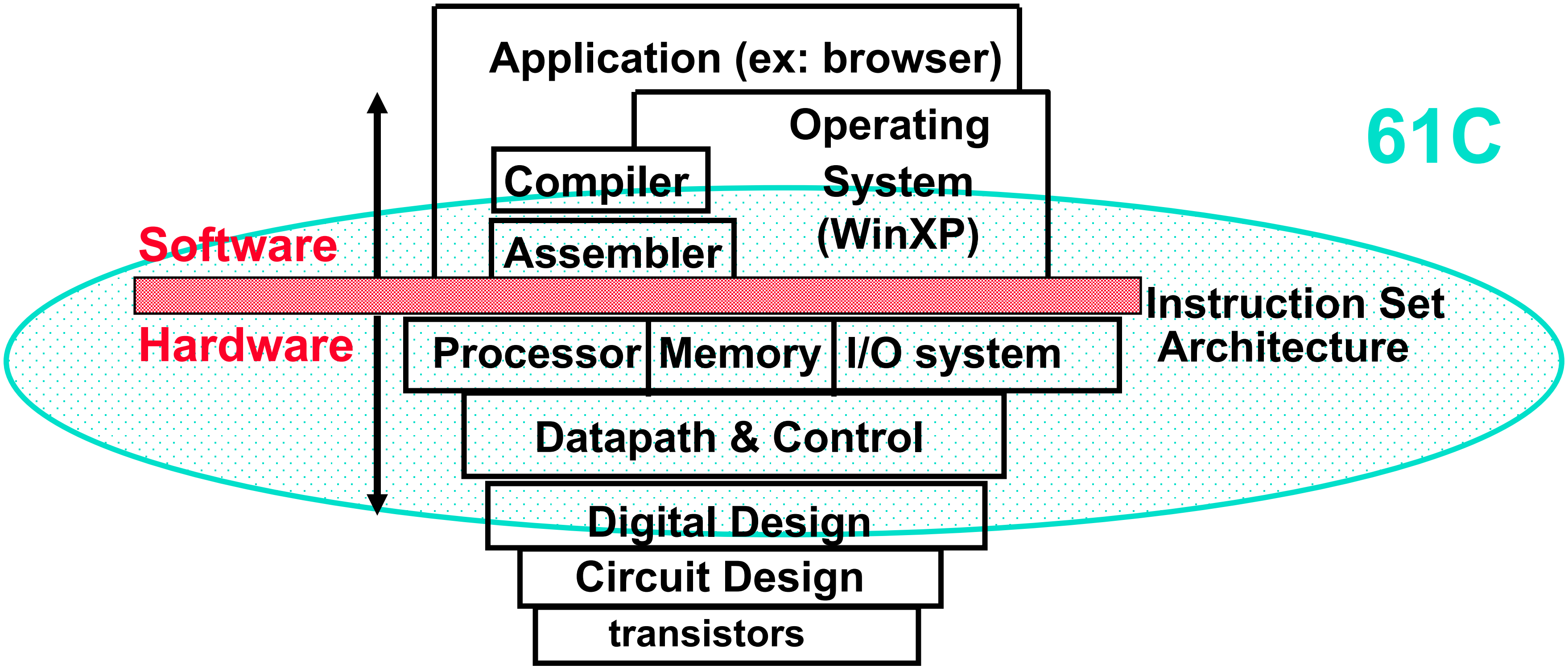
Are Computers Smart?

- To a programmer:
  - Very complex operations/functions:
    - (map (lambda (x) (* x x)) '(1 2 3 4))
  - Automatic memory management:
    - List l = new List;
  - "Basic" structures:
    - Integers, floats, characters, plus, minus, print commands

Are Computers Smart?

- In real life:
  - Only a handful of operations:
    - {and, or, not} or {nand, nor}
  - No memory management.
  - Only 2 values:
    - {0, 1} or {hi, lo} or {on, off}
    - 3 if you count <undef>

What are "Machine Structures"?



* **Coordination of many**

levels (layers) of abstraction

61C Levels of Representation

High Level Language Program (e.g., C)

temp = v[k];
v[k] = v[k+1];
v[k+1] = temp;

Compiler

Assembly Language Program (e.g.,MIPS)

```
lw   $t0, 0($2)
lw   $t1, 4($2)
sw   $t1, 0($2)
sw   $t0, 4($2)
```

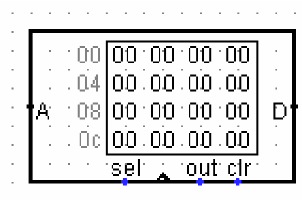
Assembler

Machine Language Program (MIPS)

```
0000 1001 1100 0110 1010 1111 0101 1000
1010 1111 0101 1000 0000 1001 1100 0110
1100 0110 1010 1111 0101 1000 0000 1001
0101 1000 0000 1001 1100 0110 1010 1111
```

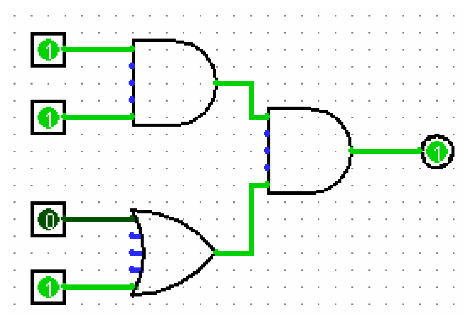
Machine Interpretation

Hardware Architecture Description (Logic, Logisim, etc.)

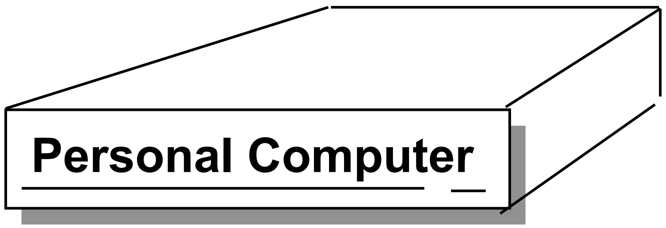


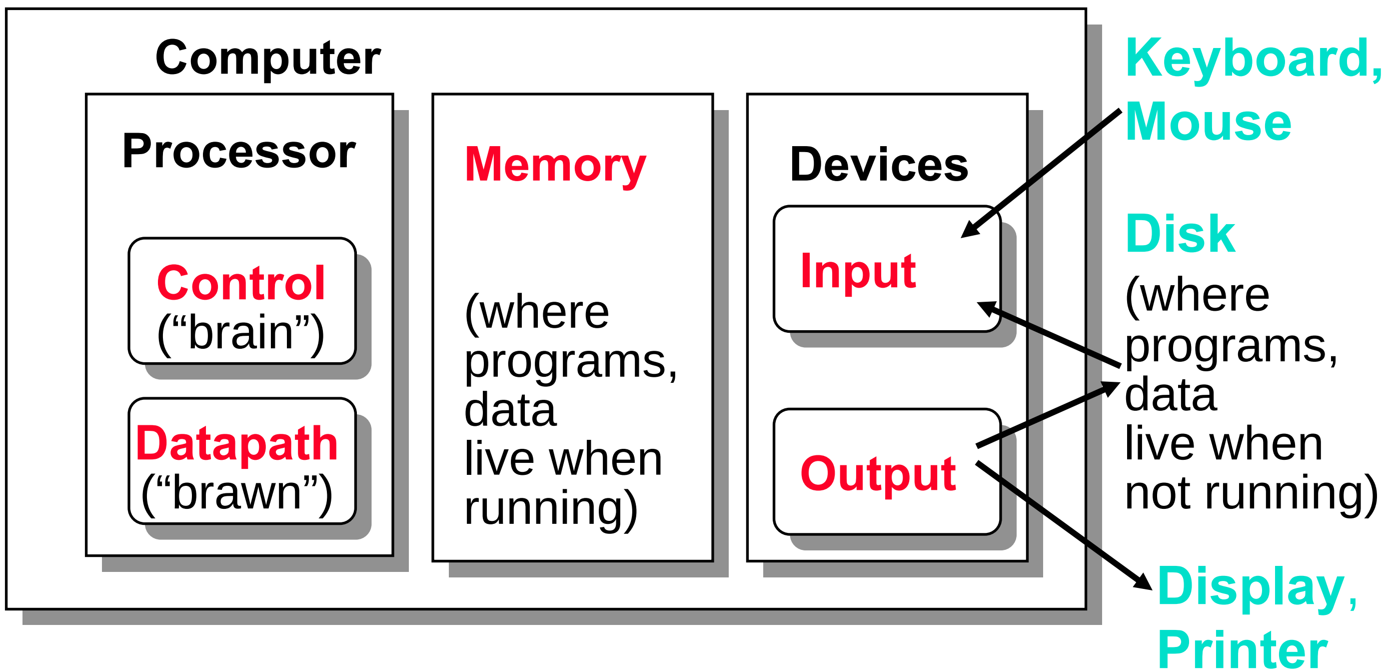
Architecture Implementation

Logic Circuit Description (Logisim, etc.)



Anatomy: 5 components of any Computer





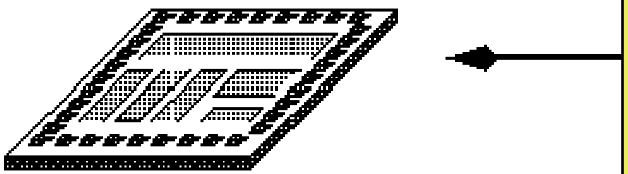
Overview of Physical Implementations

The hardware out of which we make systems.

- Integrated Circuits (ICs)
  - Combinational logic circuits, memory elements, analog interfaces.
- Printed Circuits (PC) boards
  - substrate for ICs and interconnection, distribution of CLK, Vdd, and GND signals, heat dissipation.
- Power Supplies
  - Converts line AC voltage to regulated DC low voltage levels.
- Chassis (rack, card case, ...)
  - holds boards, power supply, provides physical interface to user or other systems.
- Connectors and Cables.

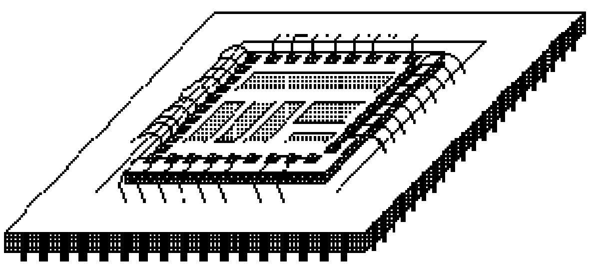
Integrated Circuits (2003 state-of-the-art)

Bare Die



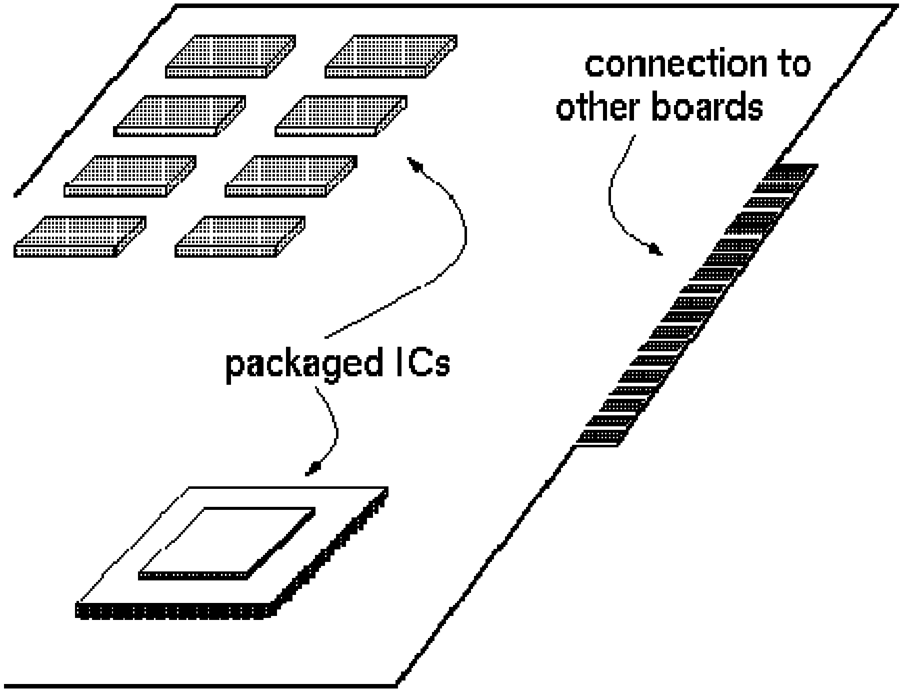
- Primarily Crystalline Silicon
- 1mm - 25mm on a side
- 2003 - feature size ~ 0.13µm = 0.13 x 10^{-6} m
- 100 - 400M transistors
- (25 - 100M “logic gates")
- 3 - 10 conductive layers
- “CMOS” (complementary metal oxide semiconductor) - most common.

Chip in Package



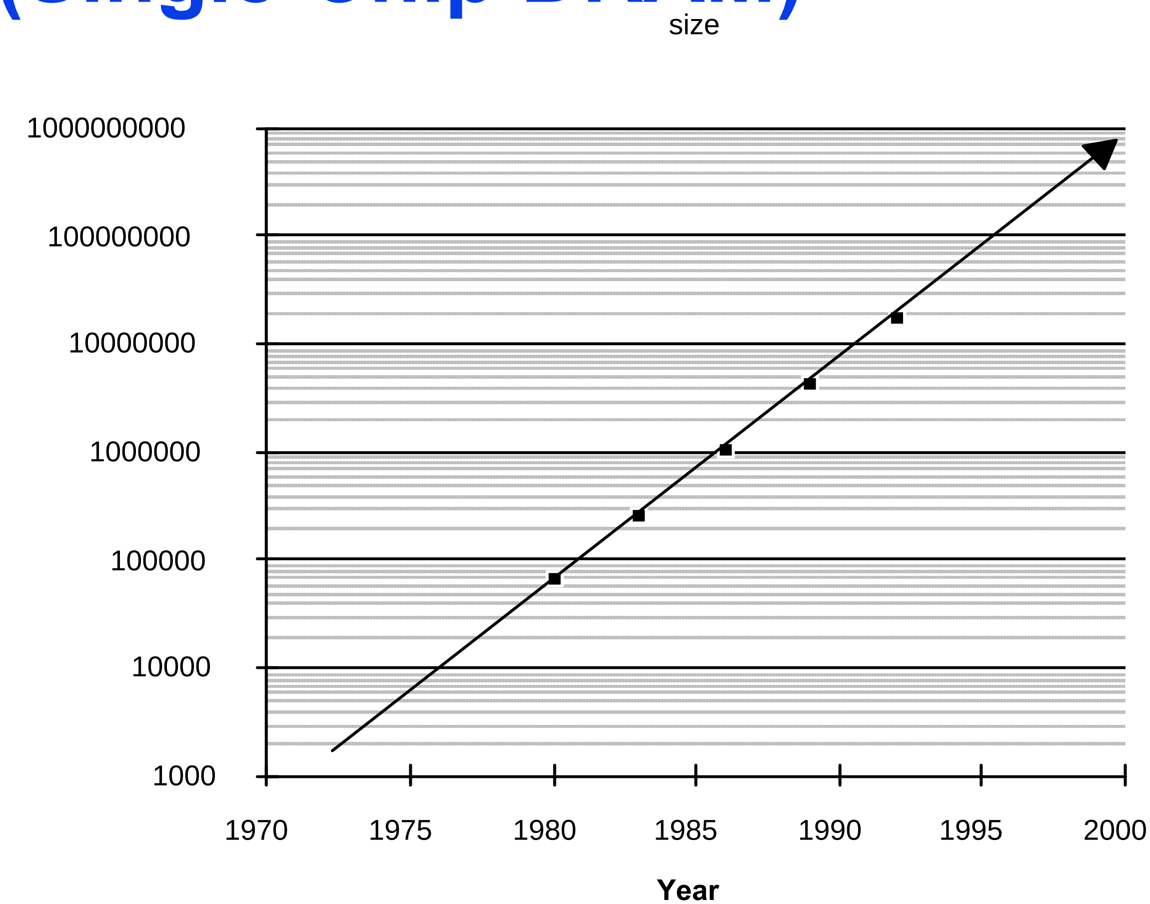
- Package provides:
  - spreading of chip-level signal paths to board-level
  - heat dissipation.
- Ceramic or plastic with gold wires.

Printed Circuit Boards



- fiberglass or ceramic
- 1-20 conductive layers
- 1-20in on a side
- IC packages are soldered down.

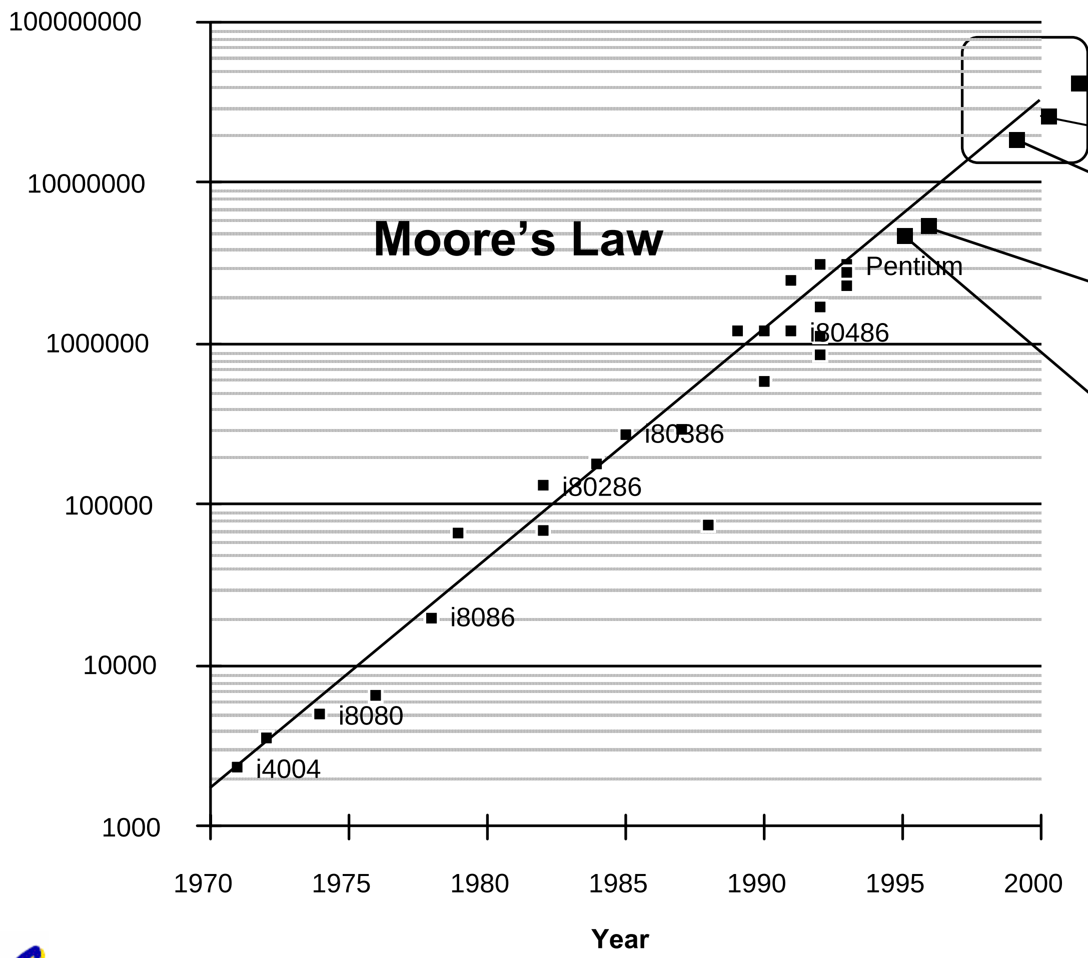
Technology Trends: Memory Capacity (Single-Chip DRAM)



year	size (Mbit)
1980	0.0625
1983	0.25
1986	1
1989	4
1992	16
1996	64
1998	128
2000	256
2002	512

- Now 1.4X/yr, or 2X every 2 years.
- 8000X since 1980!

Technology Trends: Microprocessor Complexity



Itanium 2: 41 Million

Athlon (K7): 22 Million

Alpha 21264: 15 million

Pentium Pro: 5.5 million

PowerPC 620: 6.9 million

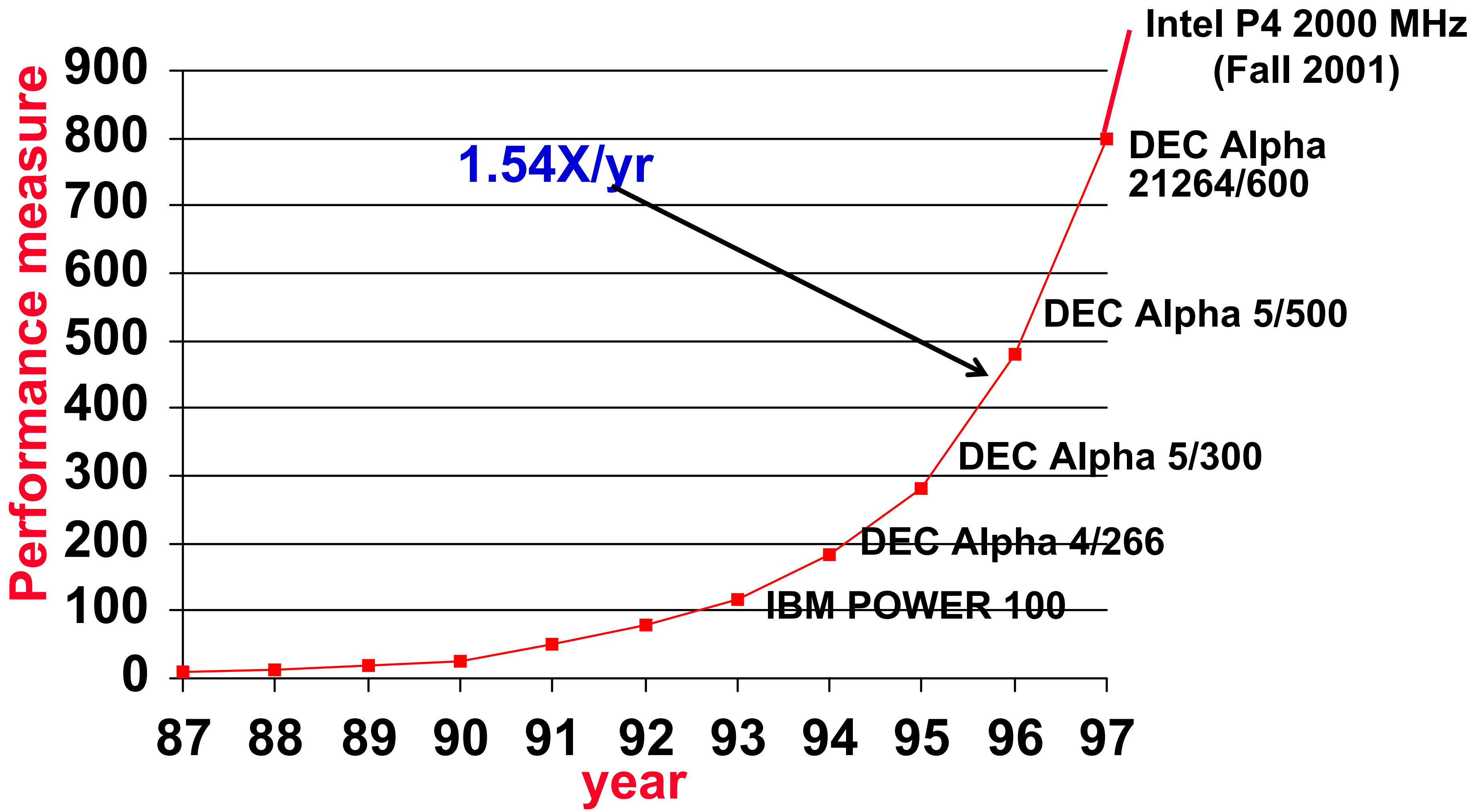
Alpha 21164: 9.3 million

Sparc Ultra: 5.2 million

2X transistors/Chip Every 1.5 years

Called "Moore's Law"

Technology Trends: Processor Performance



We'll talk about processor performance later on…

Computer Technology - Dramatic Change!

- Memory
  - DRAM capacity: 2x / 2 years (since ‘96); 64x size improvement in last decade.
- Processor
  - Speed 2x / 1.5 years (since ‘85); 100X performance in last decade.
- Disk
  - Capacity: 2x / 1 year (since ‘97) 250X size in last decade.

Computer Technology - Dramatic Change!

We'll see that Kilo, Mega, etc. are incorrect later!

- State-of-the-art PC when you graduate: (at least…)
  - Processor clock speed: 5000 MegaHertz (5.0 GigaHertz)
  - Memory capacity: 4000 MegaBytes (4.0 GigaBytes)
  - Disk capacity: 2000 GigaBytes (2.0 TeraBytes)
  - New units! Mega => Giga, Giga => Tera

    (Tera => Peta, Peta => Exa, Exa => Zetta Zetta => Yotta = 10^{24})

CS61C: So what's in it for me?

- Learn some of the big ideas in CS & engineering:
  - 5 Classic components of a Computer
  - Data can be anything (integers, floating point, characters): a program determines what it is
  - Stored program concept: instructions just data
  - Principle of Locality, exploited via a memory hierarchy (cache)
  - Greater performance by exploiting parallelism
  - Principle of abstraction, used to build systems as layers
  - Compilation v. interpretation thru system layers
  - Principles/Pitfalls of Performance Measurement

Others Skills learned in 61C

- **Learning C**
  - If you know one, you should be able to learn another programming language largely on your own
  - Given that you know C++ or Java, should be easy to pick up their ancestor, C
- **Assembly Language Programming**
  - This is a skill you will pick up, as a side effect of understanding the Big Ideas
- **Hardware design**
  - We think of hardware at the abstract level, with only a little bit of physical logic to give things perspective
  - CS 150, 152 teach this

Course Lecture Outline

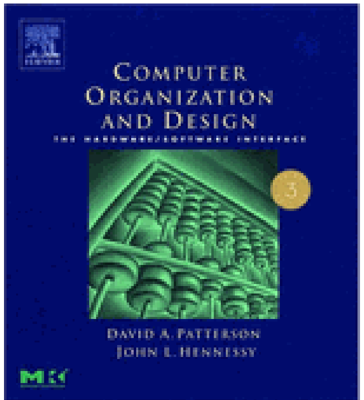
- Number representations
- C-Language (basics + pointers)
- Storage management
- Assembly Programming
- Floating Point
- `make`-ing an Executable
- Logic Design
- Introduction to Logisim
- CPU organization
- Pipelining
- Caches
- Virtual Memory
- Performance
- I/O Interrupts
- Disks, Networks
- Advanced Topics

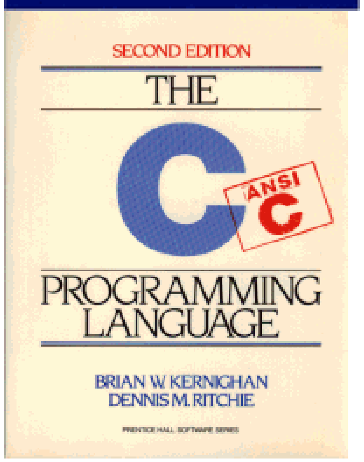
Yoda Says

Always in motion is the future…

Our schedule is very flexible. This includes lectures, assignments, exams…

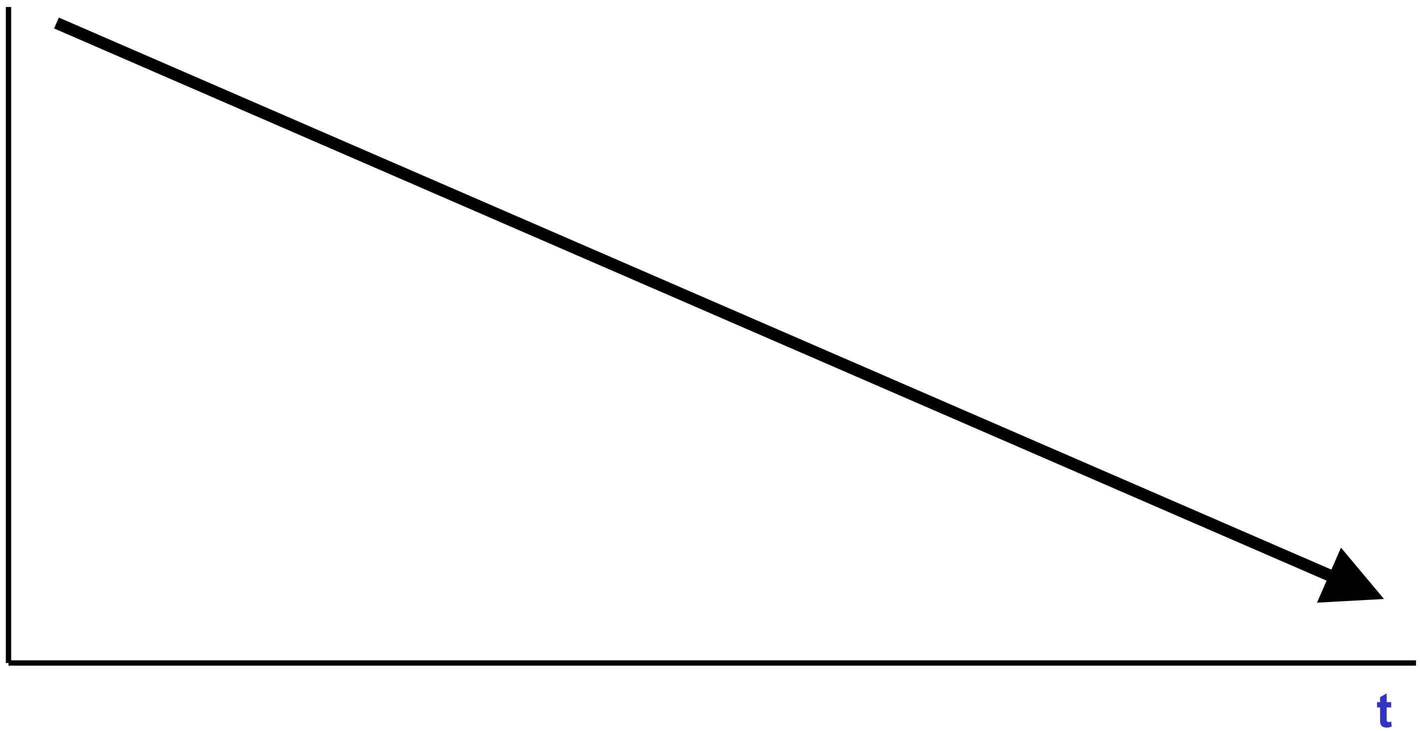
Texts





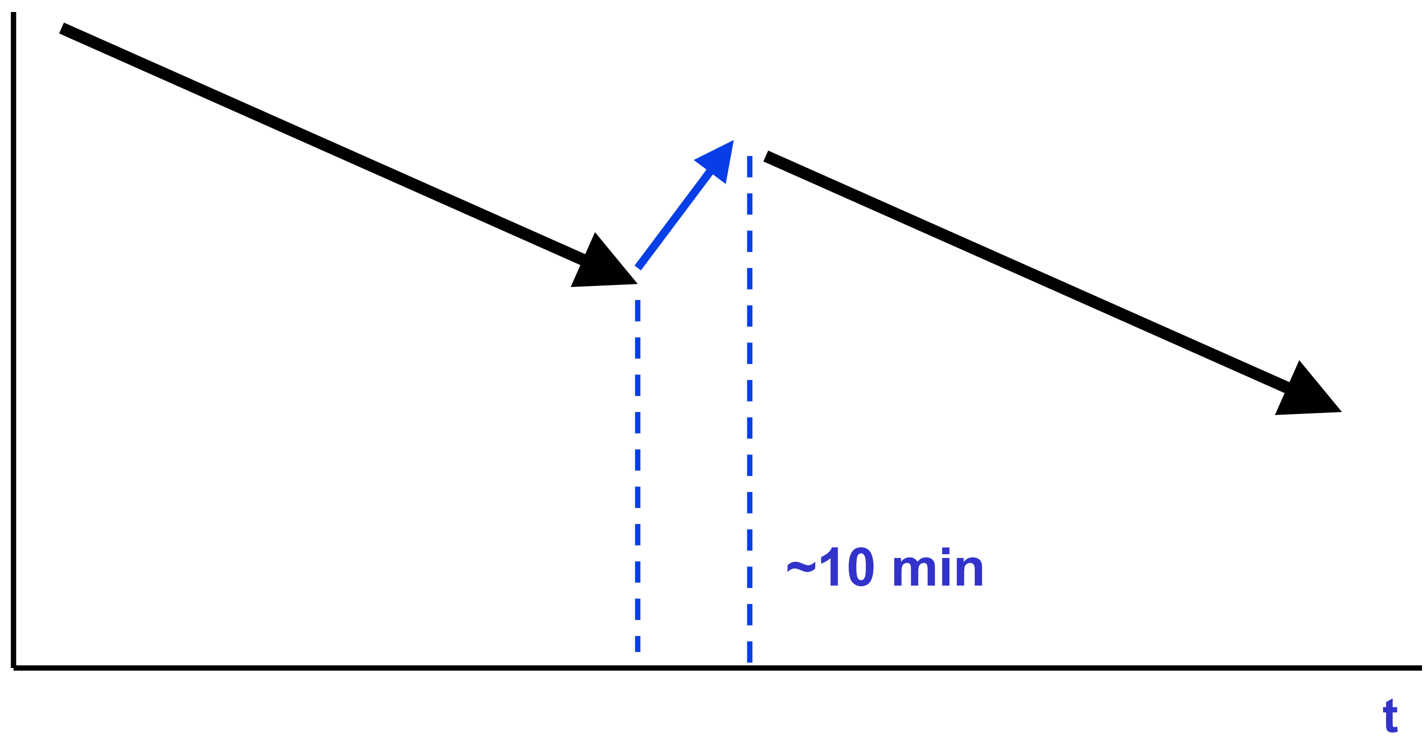
- Required: ***Computer Organization and Design: The Hardware/Software Interface, Third Edition*, Patterson and Hennessy (COD). *The second edition is far inferior, and is not suggested.***
- **Required: *The C Programming Language*, Kernighan and Ritchie (K&R), 2nd edition**
- **Reading assignments on web page**

What is this?



Attention over time!

What is this?!



Attention over time!

Administrivia

- We WILL have sections today (320 Soda)!
- HW1 is available
  - Rather simple book problems, due by the end of the day on the 26th
- Office Hours are TBD
  - But, Andy will hold a quasi office hour here after class to address any questions anyone has about the course

Assignments

- Labs
  - Mandatory – Graded on completeness
- Homework
  - Graded on correctness
- Projects
  - Graded on correctness *and* understanding
- Exams
  - Two midterms and a Final
  - Need opinions on when to schedule these

Grades

20pts	Labs
40pts	Homework
60pts	Projects (probably 4)
90pts	Midterms (2)
90pts	Final

300pts Total

Grade Scale

A+: 291 – 300	A: 271 – 290	A-: 261 – 270
B+: 251 – 260	B: 231 – 250	B-: 221 – 230
C+: 211 – 220	C: 191 – 210	C-: 180 – 190

D: 140 – 179

F: < 140

Late Assignments

- NO late homework will be accepted
  - Seriously, no late homework
- Projects may be turned in up to 24 hours late
  - But, will only be eligible for 2/3 credit
- Be aware that the instructional servers tend to slow down right around 61c deadlines
  - It is to your advantage to get assignments done early!

Cheating

- Read and understand the “Policy on Academic Honesty”
  - Available on the course website
- ASK if you have any questions about the policy
  - Ignorance of the law is not an acceptable excuse

Decimal Numbers: Base 10

Digits: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9

Example:

3271 =

$$(3 \times 10^3) + (2 \times 10^2) + (7 \times 10^1) + (1 \times 10^0)$$

Numbers: positional notation

- Number Base B ⇒ B symbols per digit:
  - Base 10 (Decimal): 0, 1, 2, 3, 4, 5, 6, 7, 8, 9
    Base 2 (Binary): 0, 1
- Number representation:
  - $d_{31}d_{30} \ldots d_1d_0$ is a 32 digit number
  - value = $d_{31} \times B^{31} + d_{30} \times B^{30} + \ldots + d_1 \times B^1 + d_0 \times B^0$
- Binary: 0,1 (In binary digits called "bits")
  - 0b11010 $= 1\times2^4 + 1\times2^3 + 0\times2^2 + 1\times2^1 + 0\times2^0$
    $= 16 + 8 + 2$
    $= 26$

#s often written 0b…

  - Here 5 digit binary # turns into a 2 digit decimal #
  - Can we find a base that converts to binary easily?

Hexadecimal Numbers: Base 16

- Hexadecimal:
  0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F
  - Normal digits + 6 more from the alphabet
  - In C, written as 0x… (e.g., 0xFAB5)
- Conversion: Binary⇔Hex
  - 1 hex digit represents 16 decimal values
  - 4 binary digits represent 16 decimal values

  ⇒1 hex digit replaces 4 binary digits
- One hex digit is a "nibble". Two is a "byte"
- Example:
  - 1010 1100 0011 (binary) = 0x_____ ?

Decimal vs. Hexadecimal vs. Binary

Examples:

1010 1100 0011 (binary)
= 0xAC3

10111 (binary)
= 0001 0111 (binary)
= 0x17

0x3F9
= 11 1111 1001 (binary)

How do we convert between hex and Decimal?

00	0	0000
01	1	0001
02	2	0010
03	3	0011
04	4	0100
05	5	0101
06	6	0110
07	7	0111
08	8	1000
09	9	1001
10	A	1010
11	B	1011
12	C	1100
13	D	1101
14	E	1110
15	F	1111

MEMORIZE!

Which base do we use?

- Decimal: great for humans, especially when doing arithmetic
- Hex: if human looking at long strings of binary numbers, its much easier to convert to hex and look 4 bits/symbol
  - Terrible for arithmetic on paper
- Binary: what computers use; you will learn how computers do +, -, *, /
  - To a computer, numbers always binary
  - Regardless of how number is written:

    32_{ten} == 32_{10} == 0x20 == 100000_2 == 0b100000
  - Use subscripts "ten", "hex", "two" in book, slides when might be confusing

What to do with representations of numbers?

- Just what we do with numbers!
  - Add them
  - Subtract them
  - Multiply them
  - Divide them
  - Compare them
- Example: 10 + 7 = 17
  - …so simple to add in binary that we can build circuits to do it!
  - subtraction just as you would in decimal
  - Comparison: How do you tell if X > Y ?

		1	1		
		1	0	1	0
+		0	1	1	1
---	---	---	---	---	---
	1	0	0	0	1

How to Represent Negative Numbers?

- So far, unsigned numbers
- Obvious solution: define leftmost bit to be sign!
  - 0 ⇒ +, 1 ⇒ -
  - Rest of bits can be numerical value of number
- This is ~ how YOU do signed numbers in decimal!
- Representation called sign and magnitude
- MIPS uses 32-bit integers. $+1_{ten}$ would be:

  0000 0000 0000 0000 0000 0000 0000 0001
- And -1_{ten} in sign and magnitude would be:

  1000 0000 0000 0000 0000 0000 0000 0001

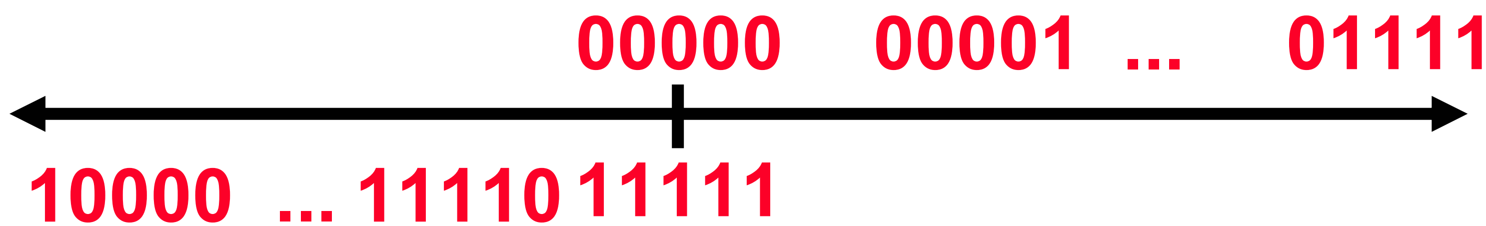
Shortcomings of sign and magnitude?

- Arithmetic circuit complicated
  - Special steps depending whether signs are the same or not
- Also, two zeros
  - 0x00000000 = $+0_{ten}$
  - 0x80000000 = -0_{ten}
  - What would two 0s mean for programming?
- Therefore sign and magnitude abandoned*

* Ask me about the star in two weeks!

Another try: complement the bits

- Example: $7_{10} = 00111_2$ $-7_{10} = 11000_2$
- Called <u>One's Complement</u>
- Note: positive numbers have leading 0s, negative numbers have leadings 1s.



- What is -00000 ? Answer: 11111
- How many positive numbers in N bits?
- How many negative ones?

Shortcomings of One's complement?

- Arithmetic still is somewhat complicated.
- Still two zeros
  - `0x00000000` = $+0_{ten}$
  - `0xFFFFFFFF` = -0_{ten}
- Although used for awhile on some computer products, one's complement was eventually abandoned because another solution was better….

Another Attempt …

- Gedanken: Decimal Car Odometer

  00003 → 00002 → 00001 → 00000 → 99999 → 99998

- Binary Odometer:

  00011 → 00010 → 00001 → 00000 → 11111 → 11110

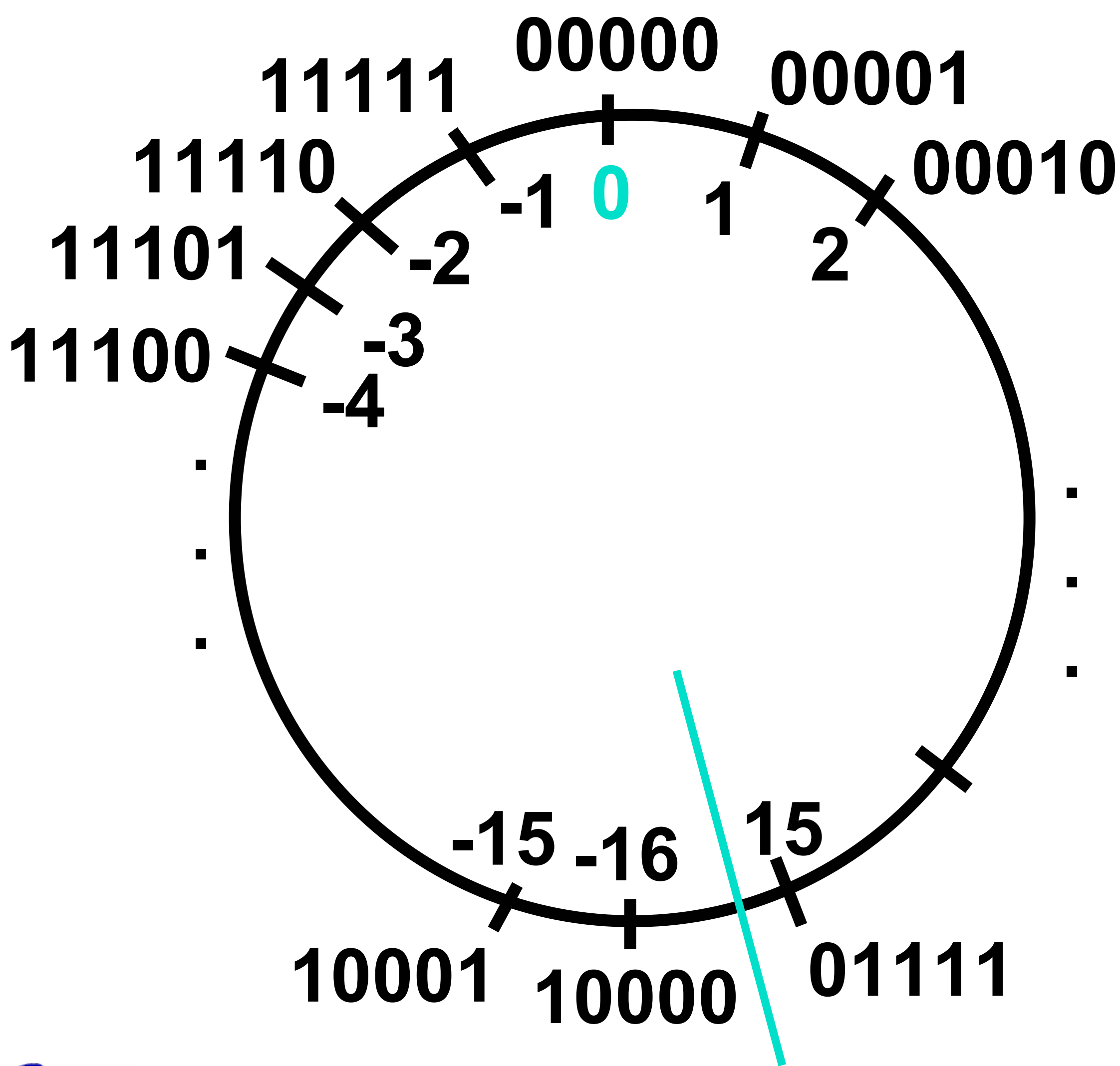
- With no obvious better alternative, pick representation that makes the math simple!
  - 99999ten == -1ten
  - 11111two == -1ten   11110two == -2ten
- This representation is Two's Complement

2's Complement Properties

- As with sign and magnitude, leading 0s ⇒ positive, leading 1s ⇒ negative
  - 000000...xxx is ≥ 0, 111111...xxx is < 0
  - except 1…1111 is -1, not -0 (as in sign & mag.)

- Only 1 Zero!

2's Complement Number "line": N = 5



- 2^{N-1} non-negatives
- 2^{N-1} negatives
- one zero
- how many positives?

Two's Complement for N=32

$0000 \ldots 0000\ 0000\ 0000\ 0000_{two} = 0_{ten}$
$0000 \ldots 0000\ 0000\ 0000\ 0001_{two} = 1_{ten}$
$0000 \ldots 0000\ 0000\ 0000\ 0010_{two} = 2_{ten}$
. . .
$0111 \ldots 1111\ 1111\ 1111\ 1101_{two} = 2{,}147{,}483{,}645_{ten}$
$0111 \ldots 1111\ 1111\ 1111\ 1110_{two} = 2{,}147{,}483{,}646_{ten}$
$0111 \ldots 1111\ 1111\ 1111\ 1111_{two} = 2{,}147{,}483{,}647_{ten}$
$1000 \ldots 0000\ 0000\ 0000\ 0000_{two} = -2{,}147{,}483{,}648_{ten}$
$1000 \ldots 0000\ 0000\ 0000\ 0001_{two} = -2{,}147{,}483{,}647_{ten}$
$1000 \ldots 0000\ 0000\ 0000\ 0010_{two} = -2{,}147{,}483{,}646_{ten}$
. . .
$1111 \ldots 1111\ 1111\ 1111\ 1101_{two} = -3_{ten}$
$1111 \ldots 1111\ 1111\ 1111\ 1110_{two} = -2_{ten}$
$1111 \ldots 1111\ 1111\ 1111\ 1111_{two} = -1_{ten}$

- One zero; 1st bit called sign bit
- 1 "extra" negative: no positive $2{,}147{,}483{,}648_{ten}$

Kilo, Mega, Giga, Tera, Peta, Exa, Zetta, Yotta

physics.nist.gov/cuu/Units/binary.html

- Common use prefixes (all SI, except K [= k in SI])

Name	Abbr	Factor	SI size
Kilo	K	2^{10} = 1,024	10^{3} = 1,000
Mega	M	2^{20} = 1,048,576	10^{6} = 1,000,000
Giga	G	2^{30} = 1,073,741,824	10^{9} = 1,000,000,000
Tera	T	2^{40} = 1,099,511,627,776	10^{12} = 1,000,000,000,000
Peta	P	2^{50} = 1,125,899,906,842,624	10^{15} = 1,000,000,000,000,000
Exa	E	2^{60} = 1,152,921,504,606,846,976	10^{18} = 1,000,000,000,000,000,000
Zetta	Z	2^{70} = 1,180,591,620,717,411,303,424	10^{21} = 1,000,000,000,000,000,000,000
Yotta	Y	2^{80} = 1,208,925,819,614,629,174,706,176	10^{24} = 1,000,000,000,000,000,000,000,000

- Confusing! Common usage of "kilobyte" means 1024 bytes, but the "correct" SI value is 1000 bytes

- Hard Disk manufacturers & Telecommunications are the only computing groups that use SI factors, so what is advertised as a 30 GB drive will actually only hold about 28 x 2^{30} bytes, and a 1 Mbit/s connection transfers 10^{6} bps.

kibi, mebi, gibi, tebi, pebi, exbi, zebi, yobi

en.wikipedia.org/wiki/Binary_prefix

- New IEC Standard Prefixes [only to exbi officially]

Name	Abbr	Factor
kibi	Ki	2^{10} = 1,024
mebi	Mi	2^{20} = 1,048,576
gibi	Gi	2^{30} = 1,073,741,824
tebi	Ti	2^{40} = 1,099,511,627,776
pebi	Pi	2^{50} = 1,125,899,906,842,624
exbi	Ei	2^{60} = 1,152,921,504,606,846,976
zebi	Zi	2^{70} = 1,180,591,620,717,411,303,424
yobi	Yi	2^{80} = 1,208,925,819,614,629,174,706,176

As of this writing, this proposal has yet to gain widespread use…

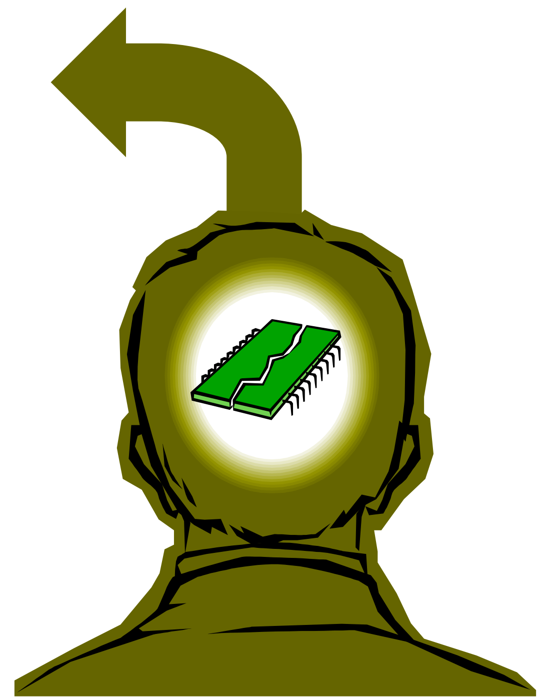
- International Electrotechnical Commission (IEC) in 1999 introduced these to specify binary quantities.
  - Names come from shortened versions of the original SI prefixes (same pronunciation) and *bi* is short for "binary", but pronounced "bee" :-(
  - Now SI prefixes only have their base-10 meaning and never have a base-2 meaning.

The way to remember #s

- What is 2^{34}? How many bits addresses (I.e., what's `ceil` $\texttt{log}_2$ = `lg` of) 2.5 TiB?
- Answer! 2^{XY} means…

X=0 ⇒ ---
X=1 ⇒ kibi ~10^3
X=2 ⇒ mebi ~10^6
X=3 ⇒ gibi ~10^9
X=4 ⇒ tebi ~10^{12}
X=5 ⇒ tebi ~10^{15}
X=6 ⇒ exbi ~10^{18}
X=7 ⇒ zebi ~10^{21}
X=8 ⇒ yobi ~10^{24}

Y=0 ⇒ 1
Y=1 ⇒ 2
Y=2 ⇒ 4
Y=3 ⇒ 8
Y=4 ⇒ 16
Y=5 ⇒ 32
Y=6 ⇒ 64
Y=7 ⇒ 128
Y=8 ⇒ 256
Y=9 ⇒ 512



MEMORIZE!