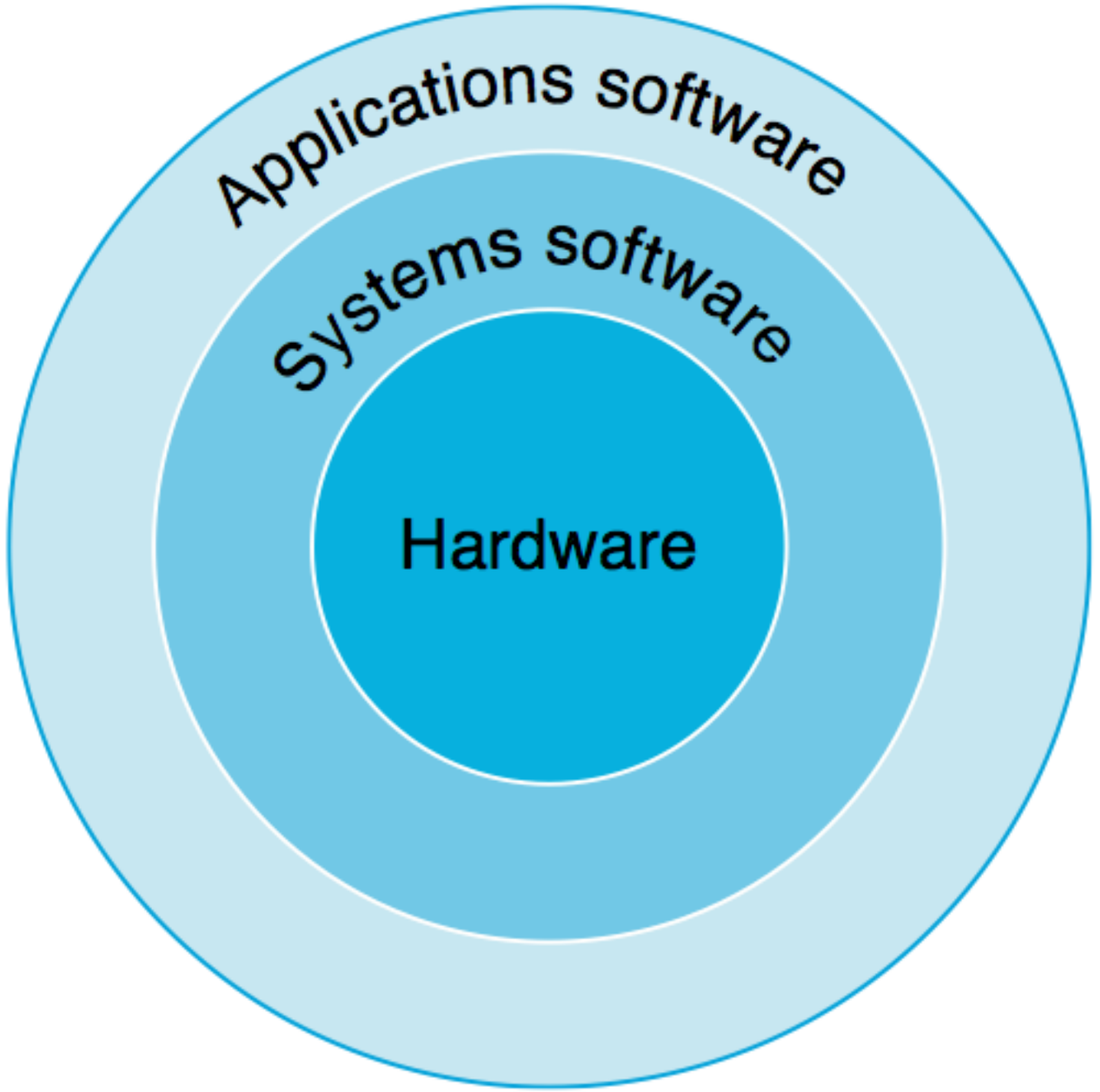
The Fundamentals of Computer Architecture



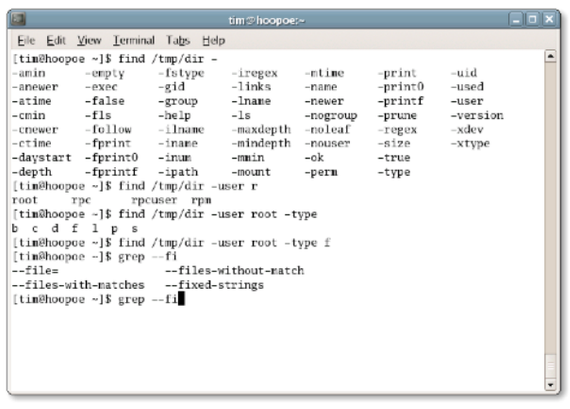
Application Software

Microsoft Office

chrome

emacs

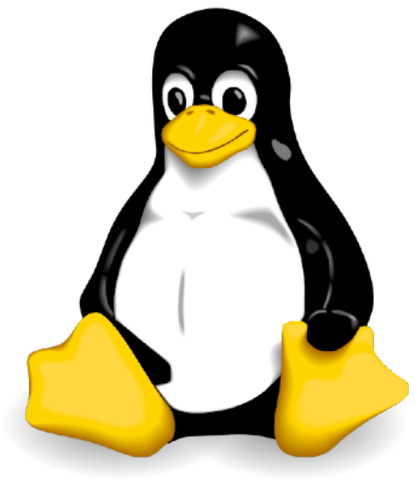
Firefox



APACHE
HTTP SERVER

Systems Software

Operating Systems

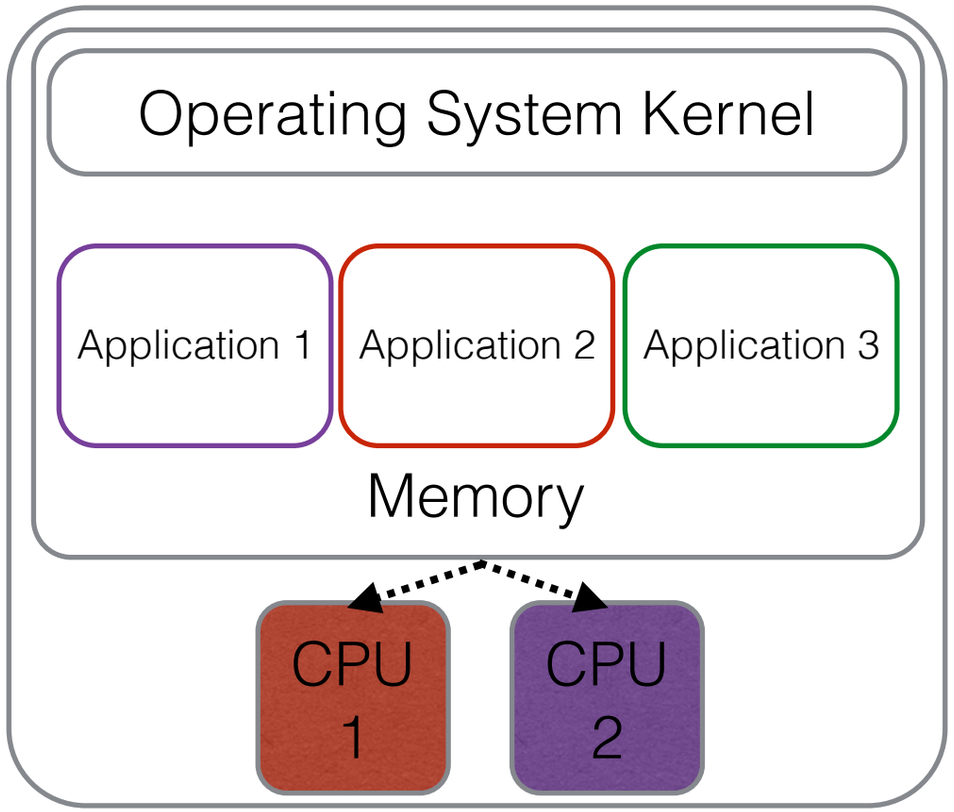


Compilers, Linkers, Assemblers

GCC:
GNU Compiler
Collection

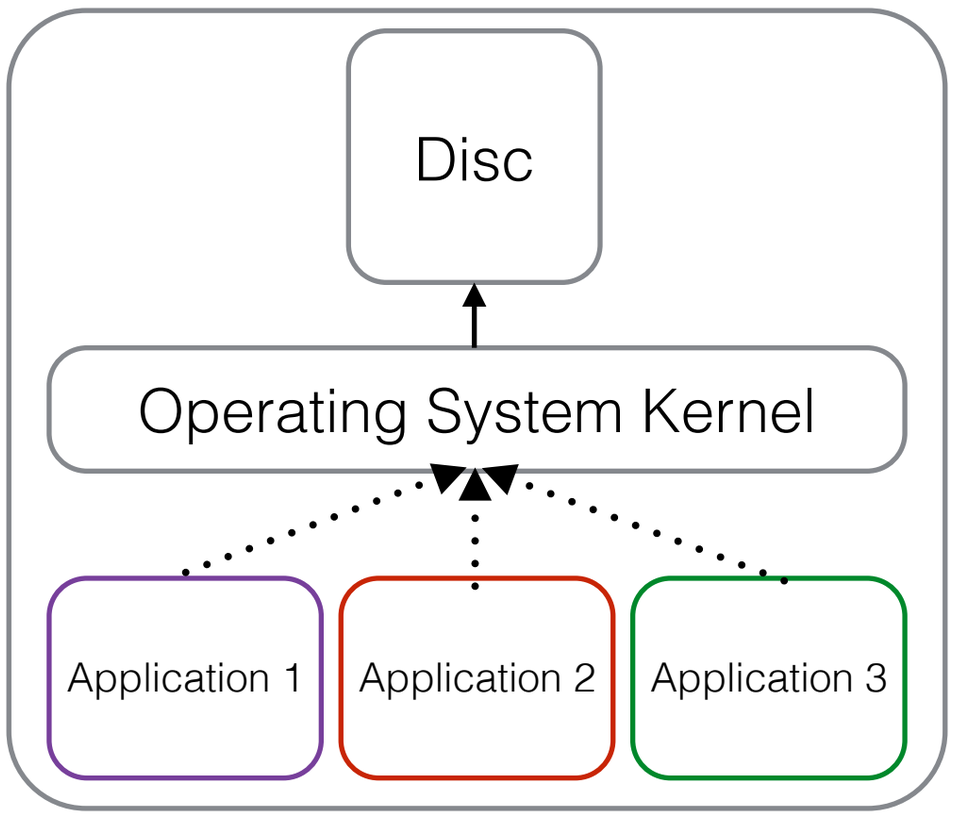
Systems Software

Operating Systems, ctd.



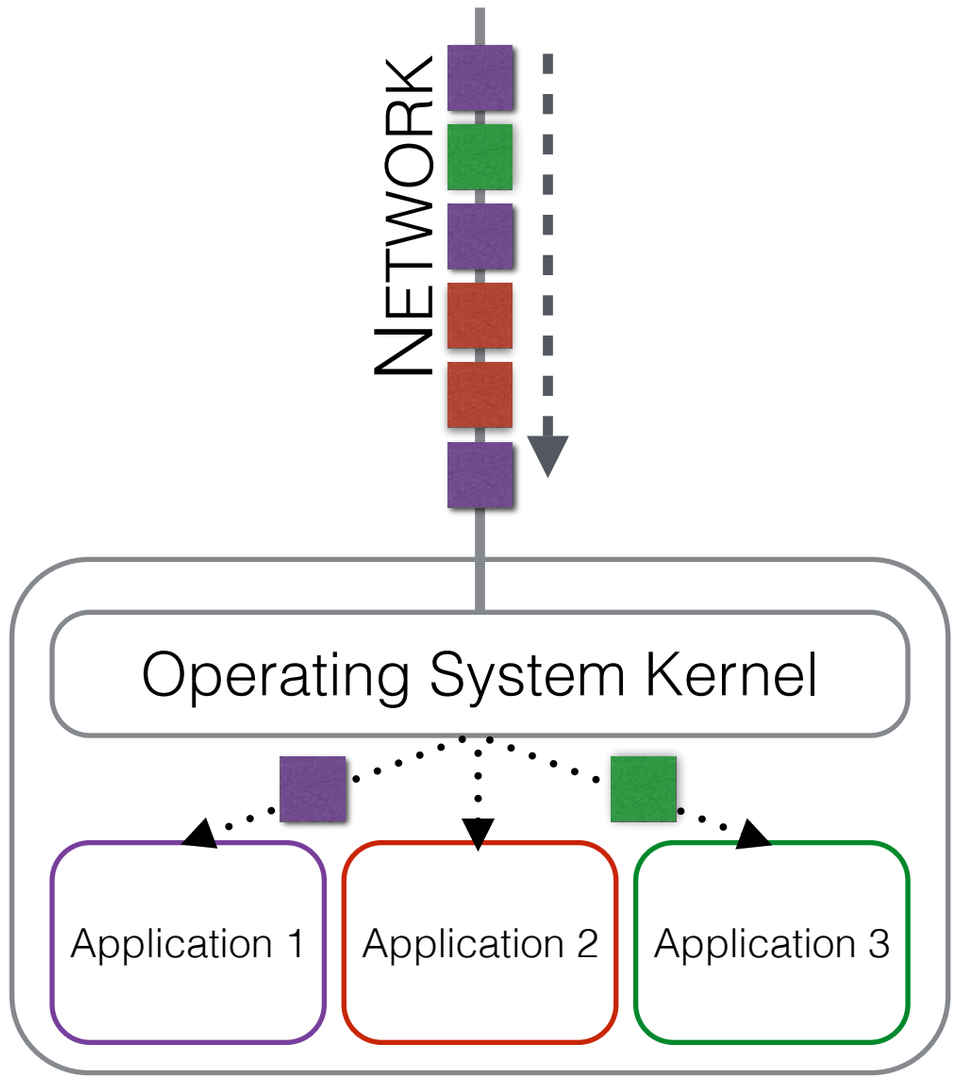
Systems Software

Operating Systems, ctd.



Systems Software

Operating Systems, ctd.



The kernel provides a common environment for applications

Systems Software

Compilers, Linkers, Assemblers

High-level Language

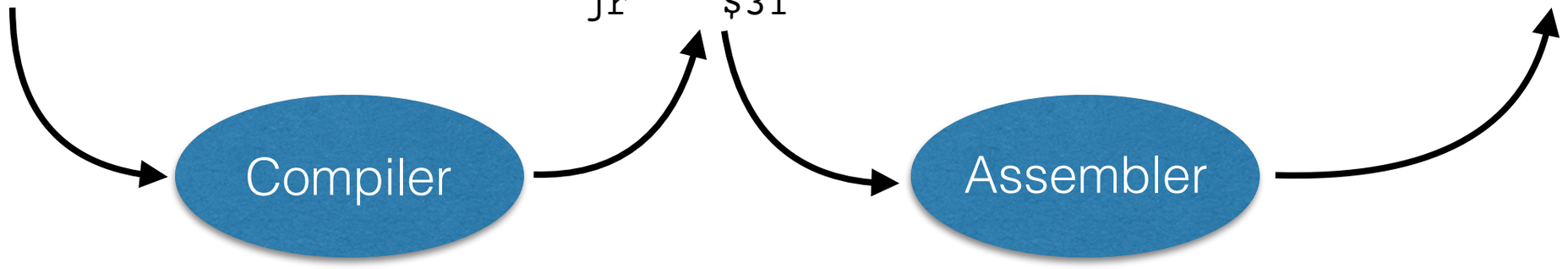
```
swap(int v[], int k)
{
     int temp;
     temp    = v[k];
     v[k]    = v[k+1];
     v[k+1] = temp;
}
```

Assembly Language

```
swap:
     multi $2, $5,4
     add   $2, $4,$2
     lw    $15, 0($2)
     lw    $16, 0($2)
     sw    $16, 0($2)
     sw    $15, 4($2)
     jr    $31
```

Machine Language

```
00000000101000100000000100011000
00000000100000100001000001000010
10001101111000100000000000000000
10001111000010010000000000000100
10101111000010010000000000000000
10101101111000100000000000000100
00000011111000000000000000001000
```

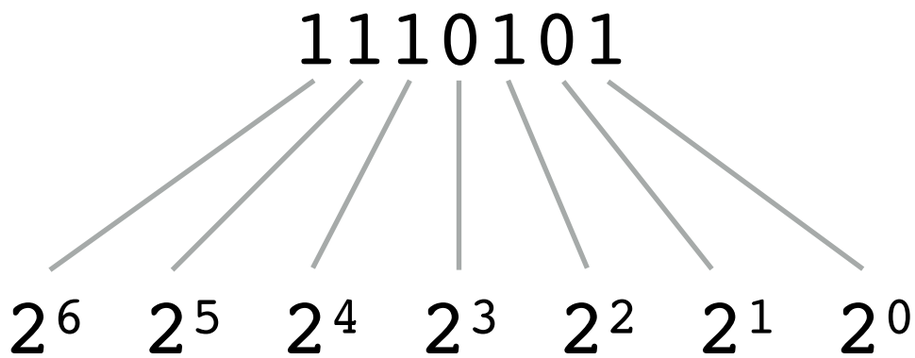


Compilers abstract the hardware

Base 10 (decimal) versus Base 2 (binary)

0	0
1	1
2	10
3	11
4	100
5	101
6	110
7	111
8	1000
9	1001
10	1010
11	1011
12	1100
13	1101
14	1110
15	1111
16	10000
17	10001
18	10010
19	10011
20	10100
...	...
99	1100011
100	1100100
101	1100101

Base 10 (decimal) versus Base 2 (binary)



Convert to Decimal:

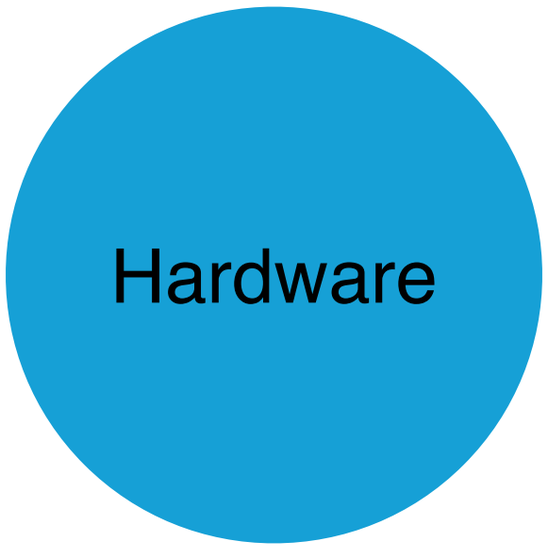
$$1x2^6 + 1x2^5 + 1x2^4 + 0x2^3 + 1x2^2 + 0x2^1 + 1x2^0$$

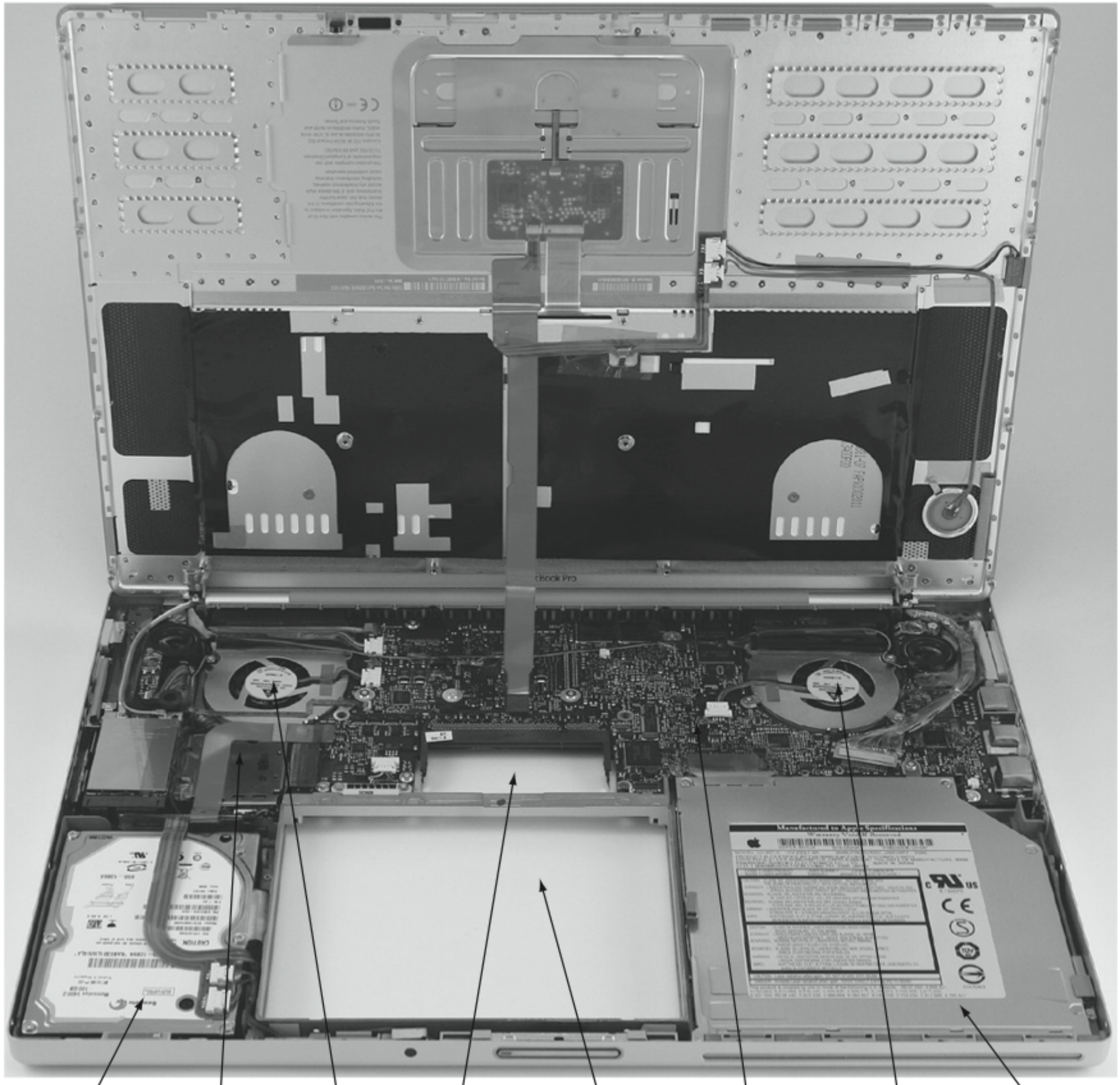
$$64 + 32 + 16 + 0 + 4 + 0 + 1$$

$$117$$

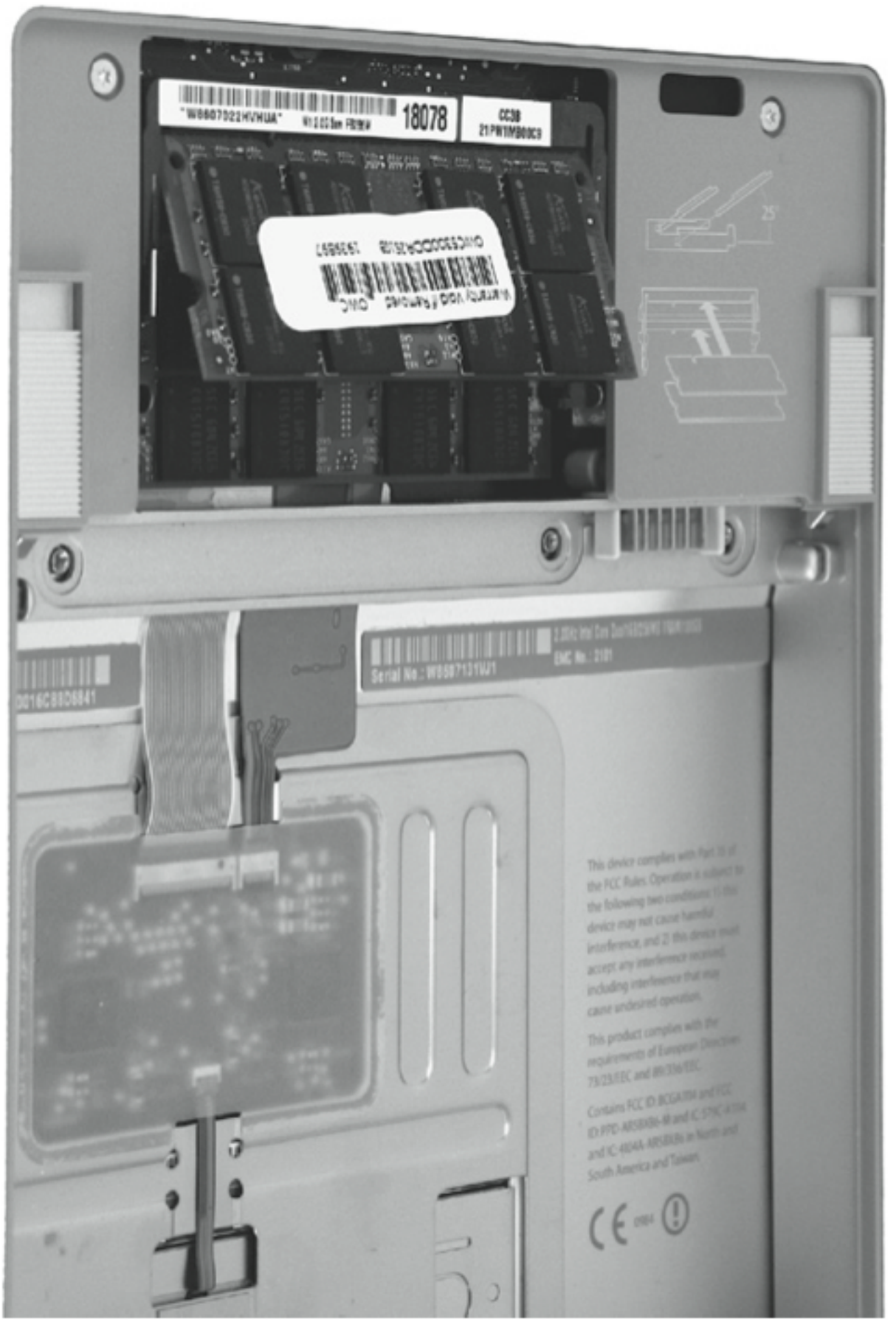
Base 10 (decimal) versus Base 2 (binary)

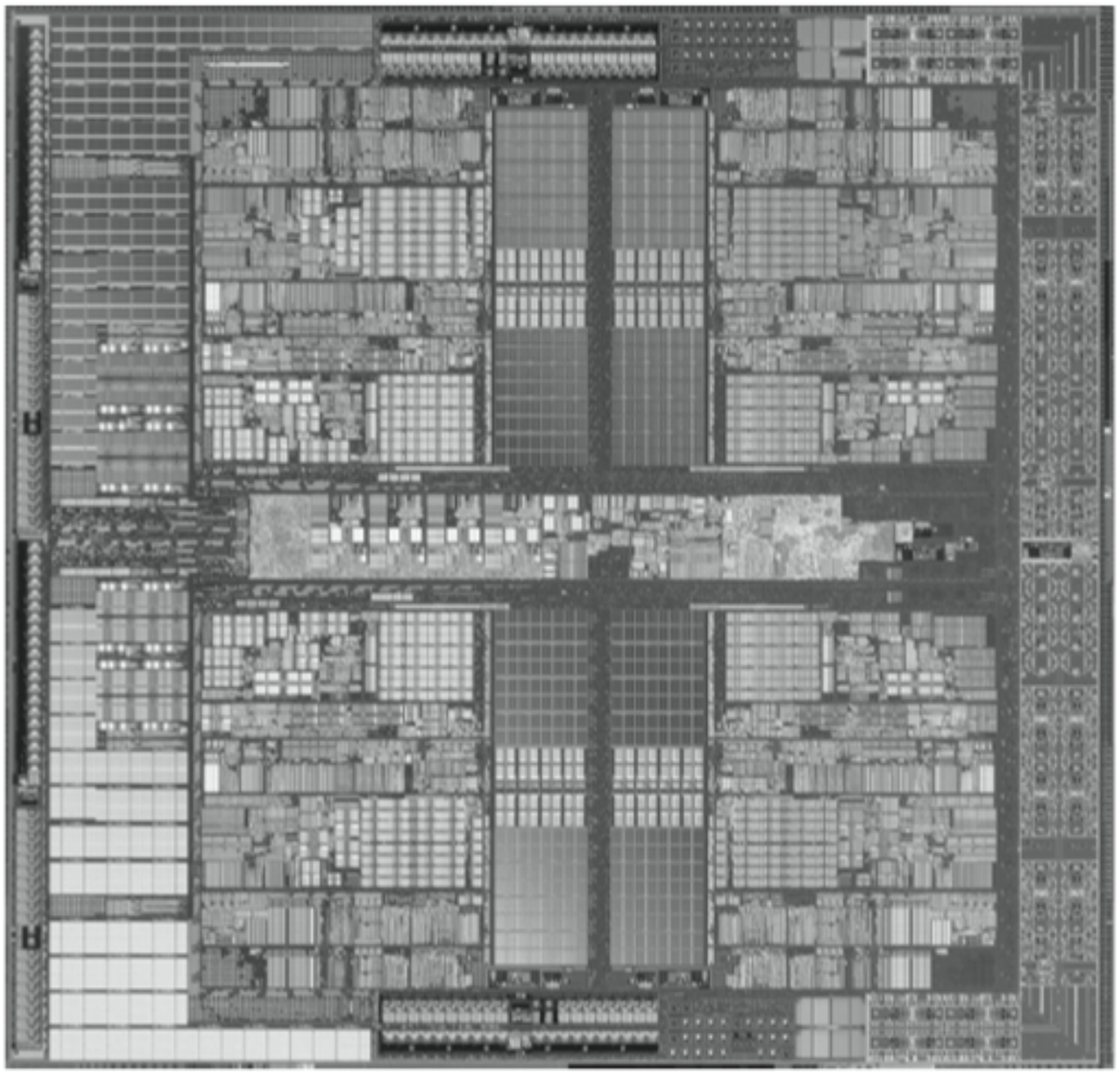
0	00000000000000000000000000000000
1	00000000000000000000000000000001
2	00000000000000000000000000000010
3	00000000000000000000000000000011
4	00000000000000000000000000000100
5	00000000000000000000000000000101
6	00000000000000000000000000000110
7	00000000000000000000000000000111
8	00000000000000000000000000001000
9	00000000000000000000000000001001
10	00000000000000000000000000001010
11	00000000000000000000000000001011
12	00000000000000000000000000001100
13	00000000000000000000000000001101
14	00000000000000000000000000001110
15	00000000000000000000000000001111
16	00000000000000000000000000010000
17	00000000000000000000000000010001
18	00000000000000000000000000010010
19	00000000000000000000000000010011
20	00000000000000000000000000010100
...	.....
99	00000000000000000000000001100011
100	00000000000000000000000001100100
101	00000000000000000000000001100101

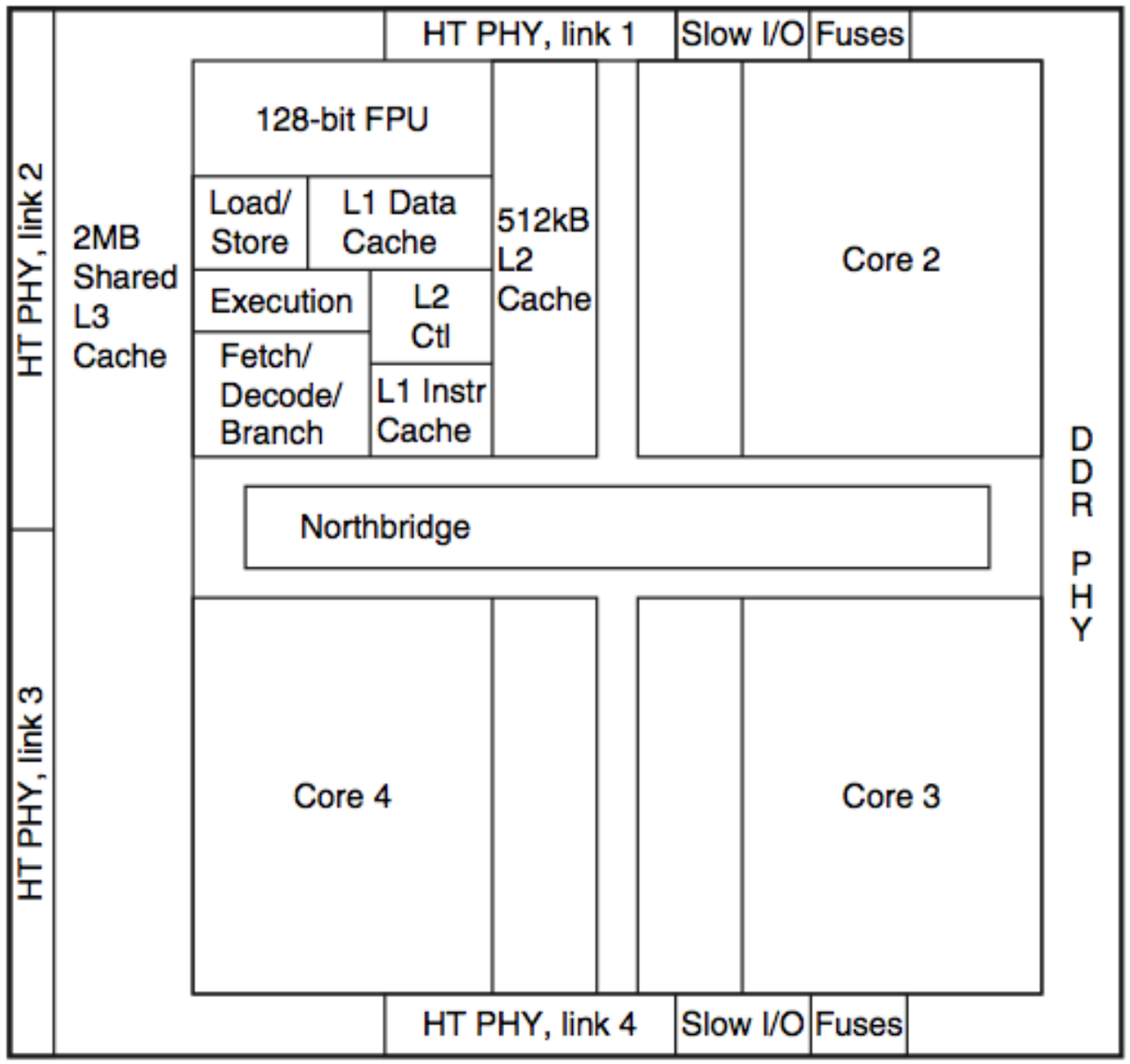

Hardware


Hard drive
Processor
Fan with cover
Spot for memory DIMMs
Spot for battery
Motherboard
Fan with cover
DVD drive

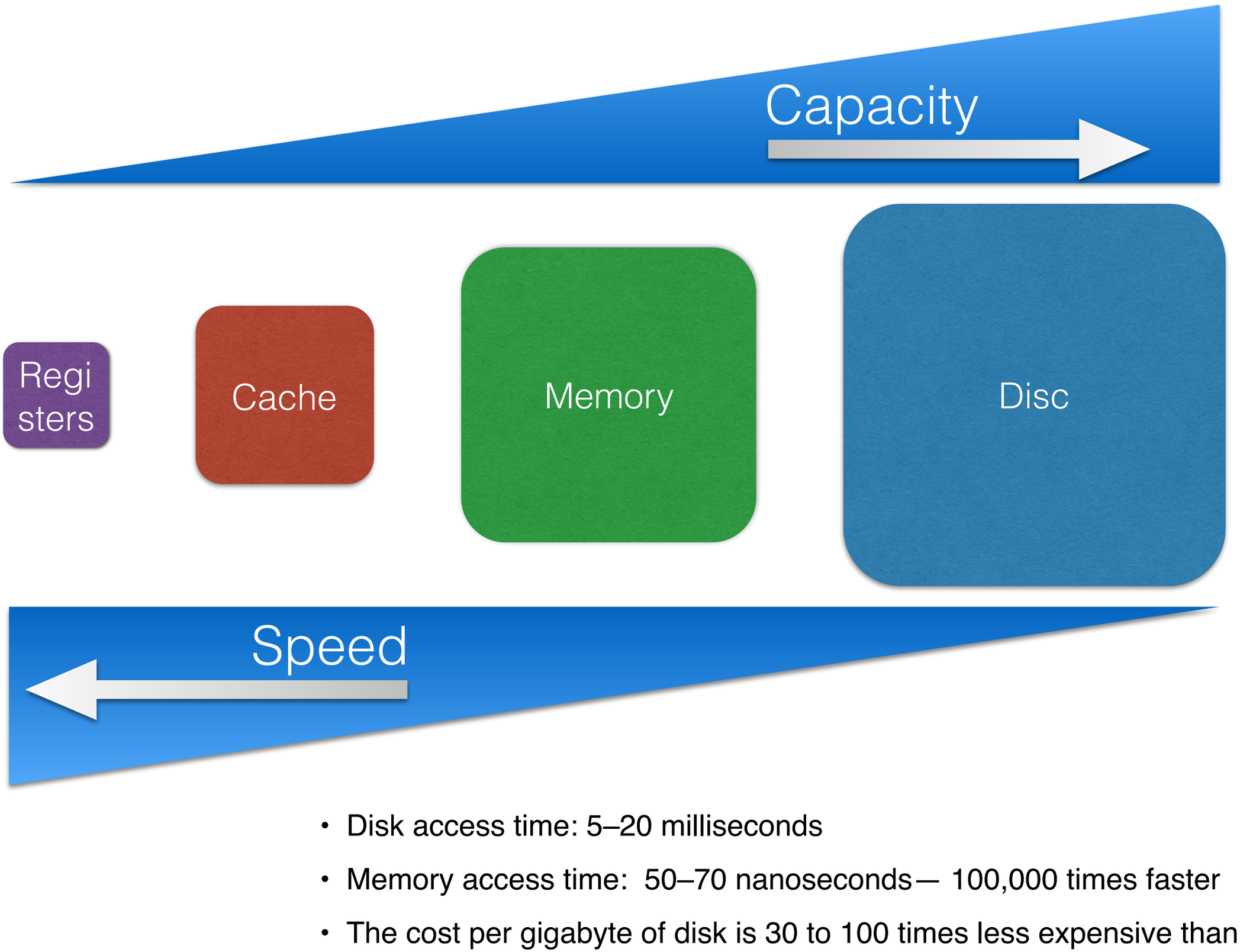
http://www.theverge.com/2011/11/23/2582677/thailand-flood-seagate-hard-drive-shortage


18078
CC38
Serial No : W8807131WJ1
This device complies with Part 15 of the FCC Rules. Operation is subject to the following two conditions: (1) this device may not cause harmful interference, and (2) this device must accept any interference received, including interference that may cause undesired operation.
This product complies with the requirements of European Directives 73/23/EEC and 89/336/EEC.
Contains FCC ID: BCGA1114 and FCC ID: PPD-AR5008x-M and IC: 579C-4114 and IC: 4104A-AR5008x in North and South America and Taiwan.

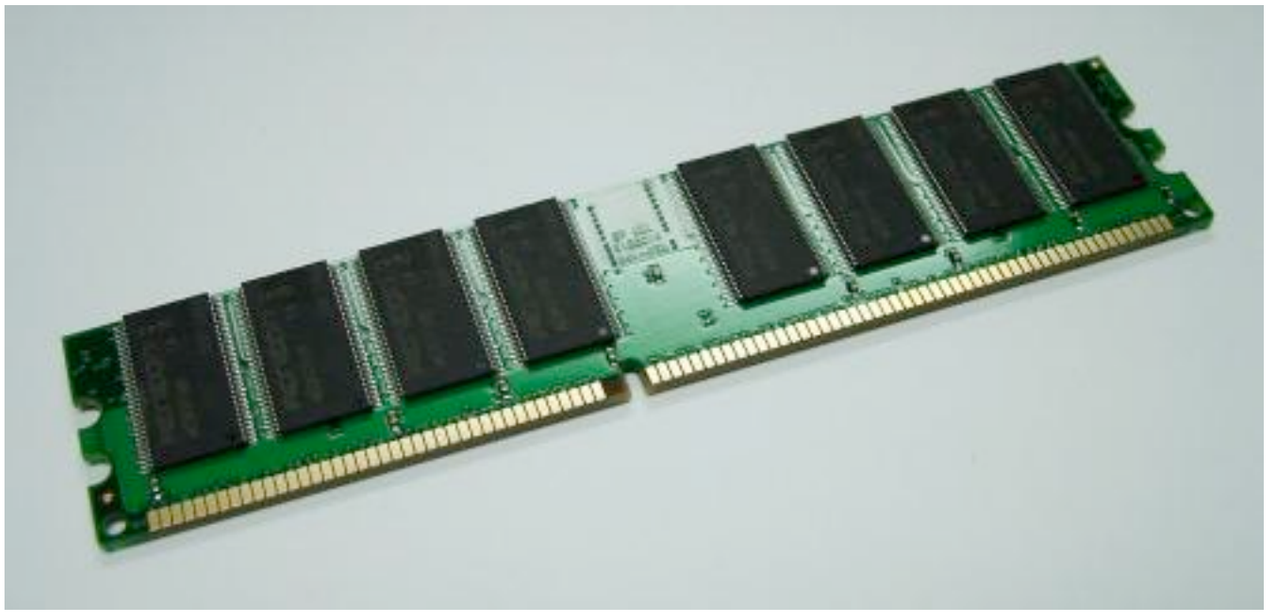


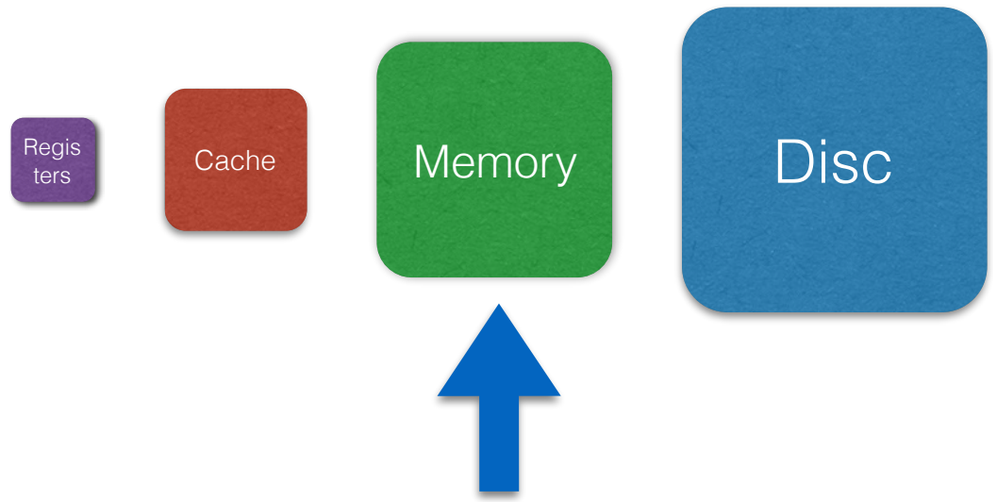

HT PHY, link 1
Slow I/O
Fuses
128-bit FPU
Load/ Store
L1 Data Cache
Execution
L2 Ctl
Fetch/ Decode/ Branch
L1 Instr Cache
512kB L2 Cache
Core 2
HT PHY, link 2
2MB Shared L3 Cache
DDR PHY
Northbridge
Core 4
Core 3
HT PHY, link 3
HT PHY, link 4
Slow I/O
Fuses

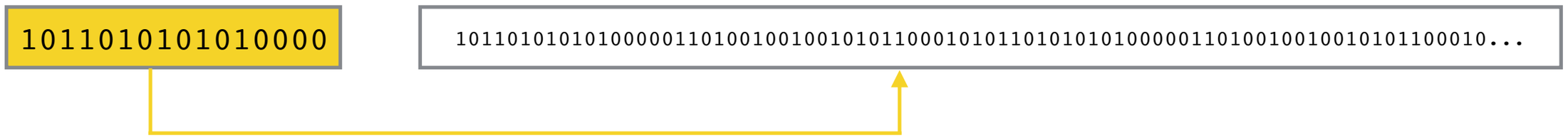
Storing code and data



- Disk access time: 5–20 milliseconds
- Memory access time: 50–70 nanoseconds— 100,000 times faster
- The cost per gigabyte of disk is 30 to 100 times less expensive than memory





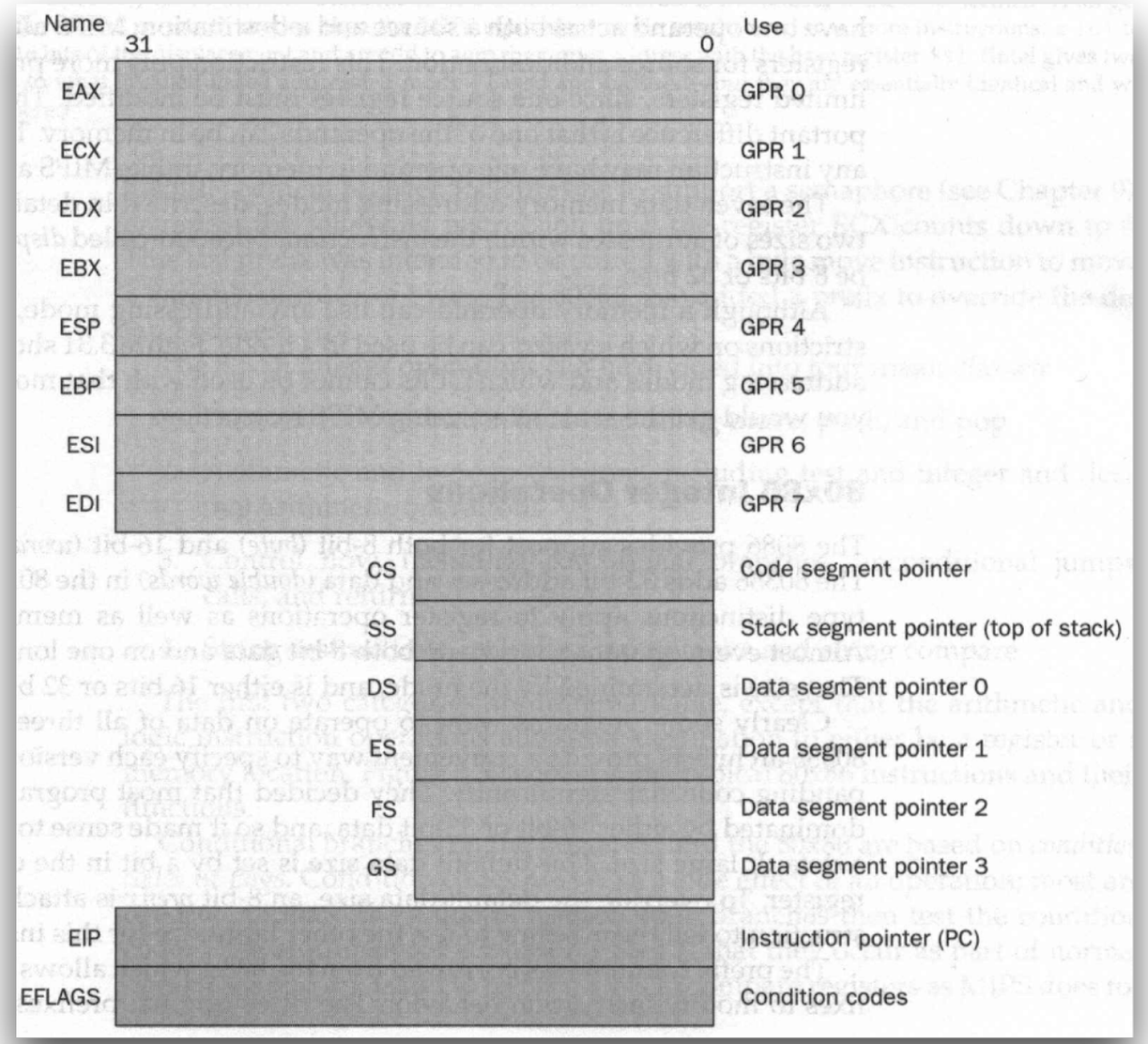


Memory: RAM - Random Access Memory

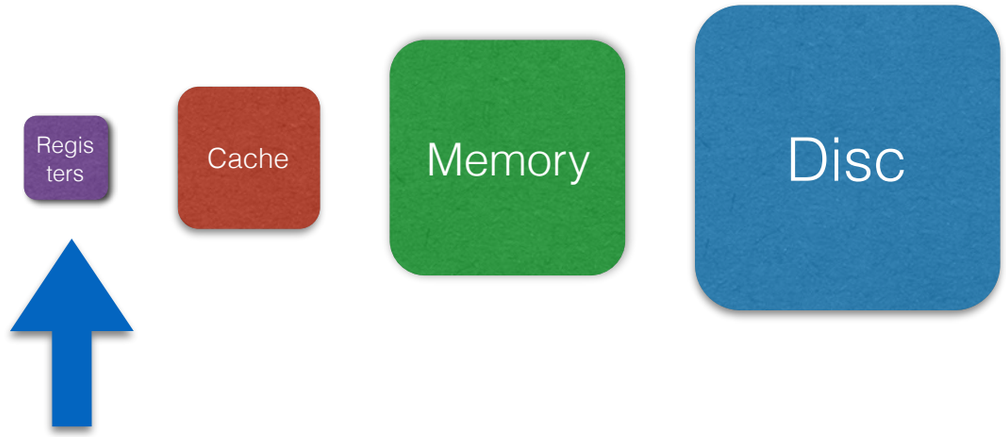
32-bit computer: address fields are 32 bits wide

allows 2^{32} possible addresses, or 4Gb

machines are now 64-bit, allowing memories up to 2^{64}



http://www.cs.mcgill.ca/~jvybihal/Courses/CS273/8086%20Registers.JPG




```
10110101010100001011010101010000
```

Registers have a fixed size and a fixed count

Some registers are ‘general purpose’ some are specialized

an instruction format

Translating a MIPS Assembly Instruction into a Machine Instruction

Let's do the next step in the refinement of the MIPS language as an example. We'll show the real MIPS language version of the instruction represented symbolically as

```
add $t0,$s1,$s2
```

first as a combination of decimal numbers and then of binary numbers.

The decimal representation is

0	17	18	8	0	32

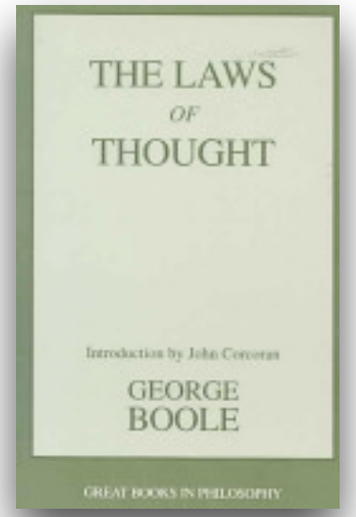
Each of these segments of an instruction is called a *field*. The first and last fields (containing 0 and 32 in this case) in combination tell the MIPS computer that this instruction performs addition. The second field gives the number of the register that is the first source operand of the addition operation (17 = `$s1`), and the third field gives the other source operand for the addition (18 = `$s2`). The fourth field contains the number of the register that is to receive the sum (8 = `$t0`). The fifth field is unused in this instruction, so it is set to 0. Thus, this instruction adds register `$s1` to register `$s2` and places the sum in register `$t0`.

This instruction can also be represented as fields of binary numbers as opposed to decimal:

000000	10001	10010	01000	00000	100000
6 bits	5 bits	5 bits	5 bits	5 bits	6 bits

Base 10 (decimal) versus Base 2 (binary)

0	00000000000000000000000000000000
1	00000000000000000000000000000001
2	00000000000000000000000000000010
3	00000000000000000000000000000011
4	00000000000000000000000000000100
5	00000000000000000000000000000101
6	00000000000000000000000000000110
7	00000000000000000000000000000111
8	00000000000000000000000000001000
9	00000000000000000000000000001001
10	00000000000000000000000000001010
11	00000000000000000000000000001011
12	00000000000000000000000000001100
13	00000000000000000000000000001101
14	00000000000000000000000000001110
15	00000000000000000000000000001111
16	00000000000000000000000000010000
17	00000000000000000000000000010001
18	00000000000000000000000000010010
19	00000000000000000000000000010011
20	00000000000000000000000000010100
...	.....
99	00000000000000000000000001100011
100	00000000000000000000000001100100
101	00000000000000000000000001100101



1854, “a Mathematical language dealing with the questions of logic”

Boolean Logic / Boolean Algebra

True = 1
False = 0

AND operation: * or &

A * B

1 * 1 = True
1 * 0 = False
0 * 1 = False
0 * 0 = False

OR operation: + or |

A + B

1 + 1 = True
1 + 0 = True
0 + 1 = True
0 + 0 = False

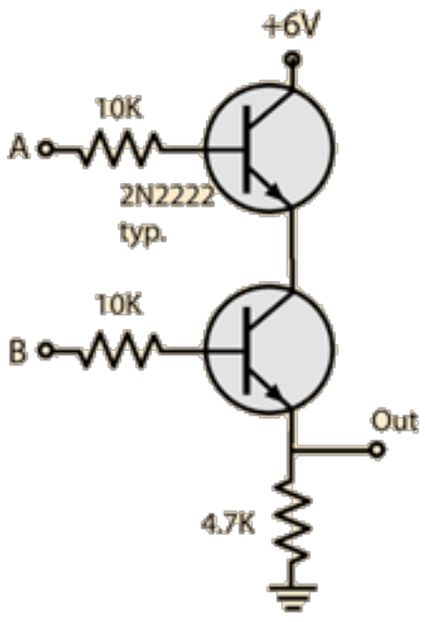
NOT operation: $^-$

$\bar{A}$

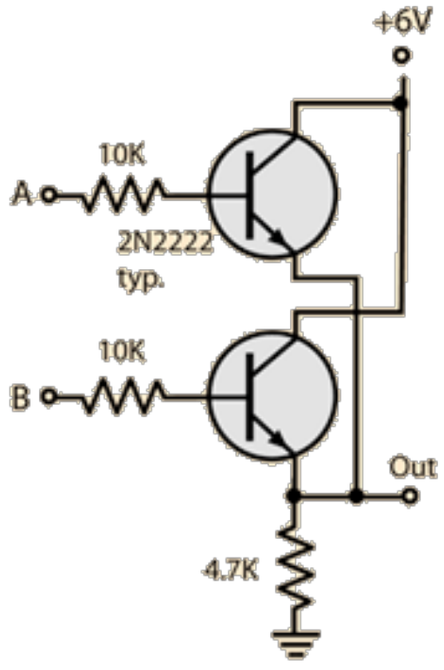
1 = False
0 = True

Boolean logic, ctd.

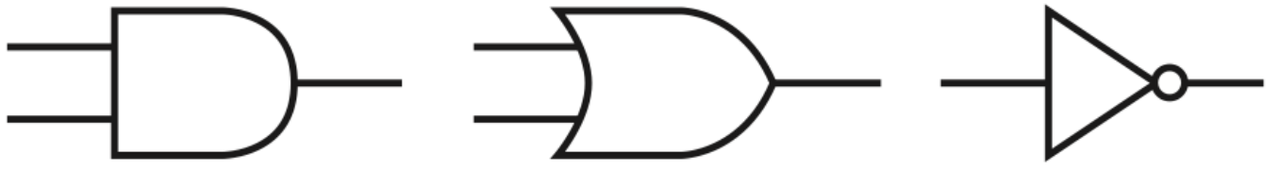
AND gate



OR gate



AND OR NOT



http://hyperphysics.phy-astr.gsu.edu/hbase/electronic/trangate.html

How does computation happen?

Addition: 6 + 7

C code

```
i = 6
j = 7
g = i + j;
```

Assembly

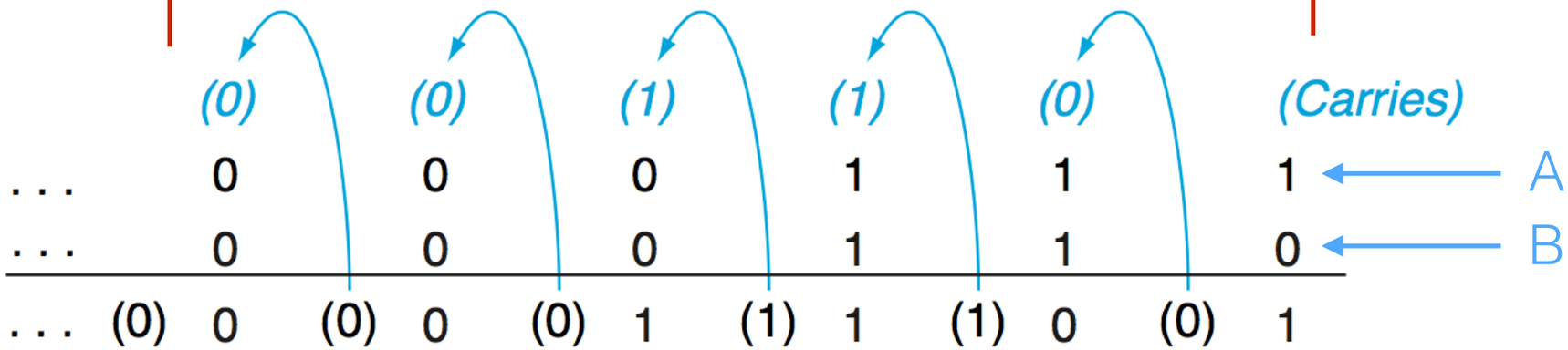
```
lw  $t0, 4($s3)
lw  $t1, 8($s3)
add $t0, $t1, $t2
sw  $t2, 48($s3)
```

Machine code

```
10100001110000100010101000001100
10101001010000000010101000001001
01100001100010000010100001101000
00011001110000010010101011001000
```

Binary Addition

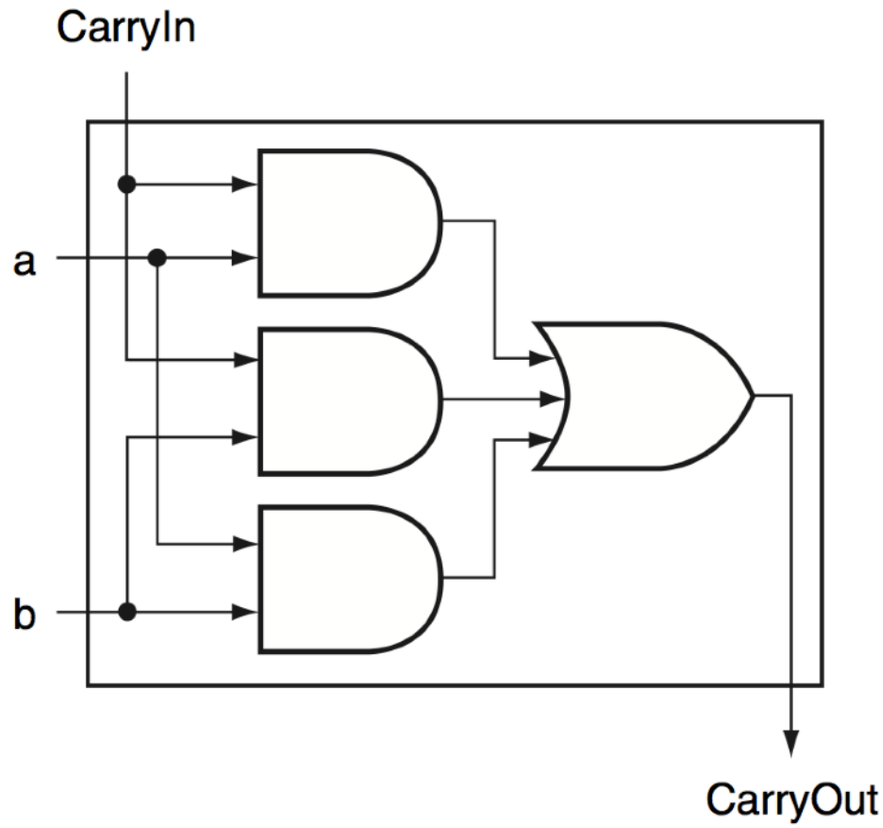
$$
\begin{array}{lll}
 & 0000\ 0000\ 0000\ 0000\ 0000\ 0000\ 0000\ 0111_{two} & = 7_{ten} \\
+ & 0000\ 0000\ 0000\ 0000\ 0000\ 0000\ 0000\ 0110_{two} & = 6_{ten} \\
\hline
= & 0000\ 0000\ 0000\ 0000\ 0000\ 0000\ 0000\ 1101_{two} & = 13_{ten}
\end{array}
$$



Inputs			Outputs	
A	B	CarryIn	CarryOut	Sum
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1

Binary Addition, ctd.

Inputs			Outputs	
A	B	CarryIn	CarryOut	Sum
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1



CarryOut = (b * CarryIn) + (a * CarryIn) + (a * b)

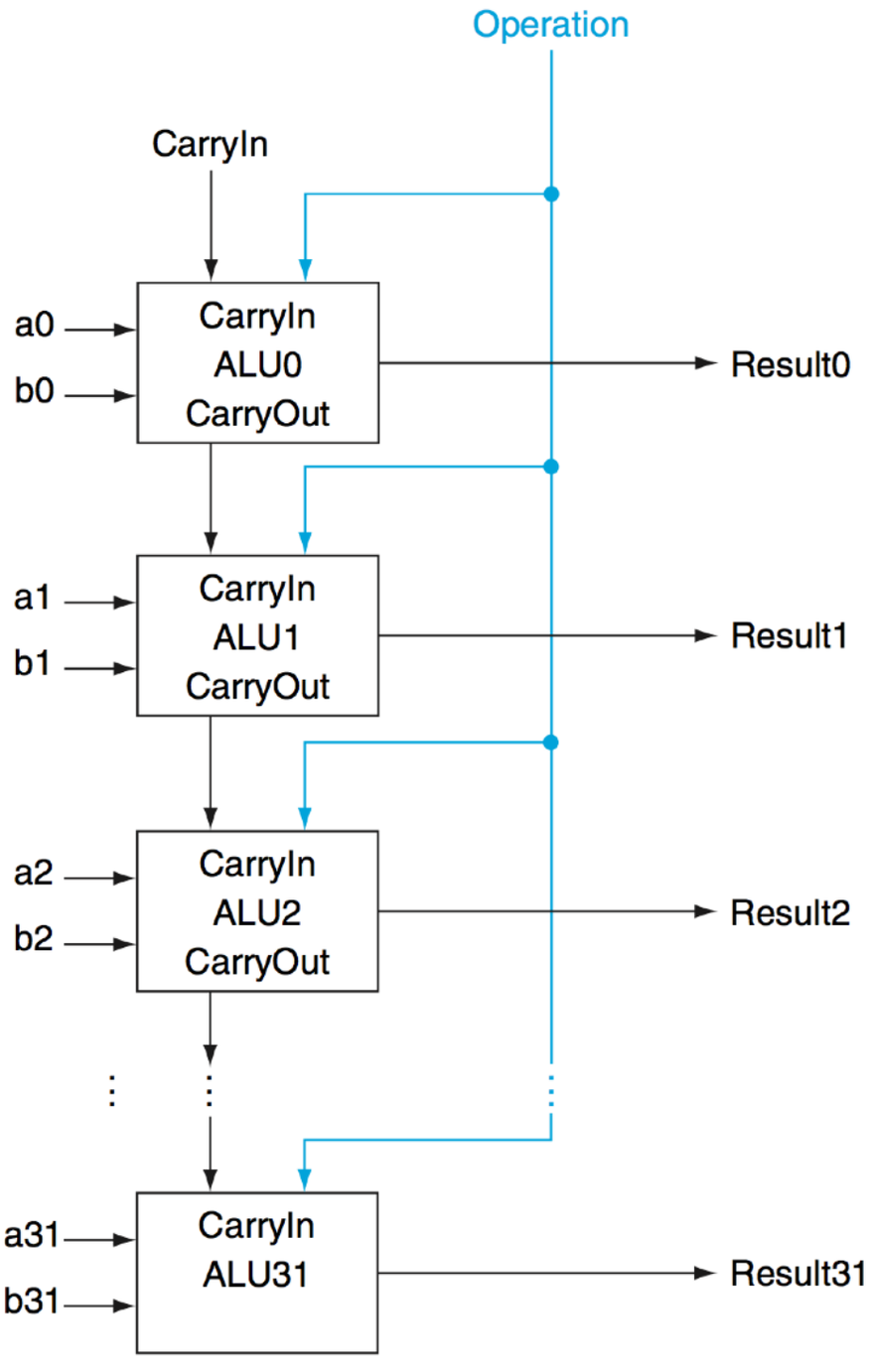
Binary Addition, ctd.

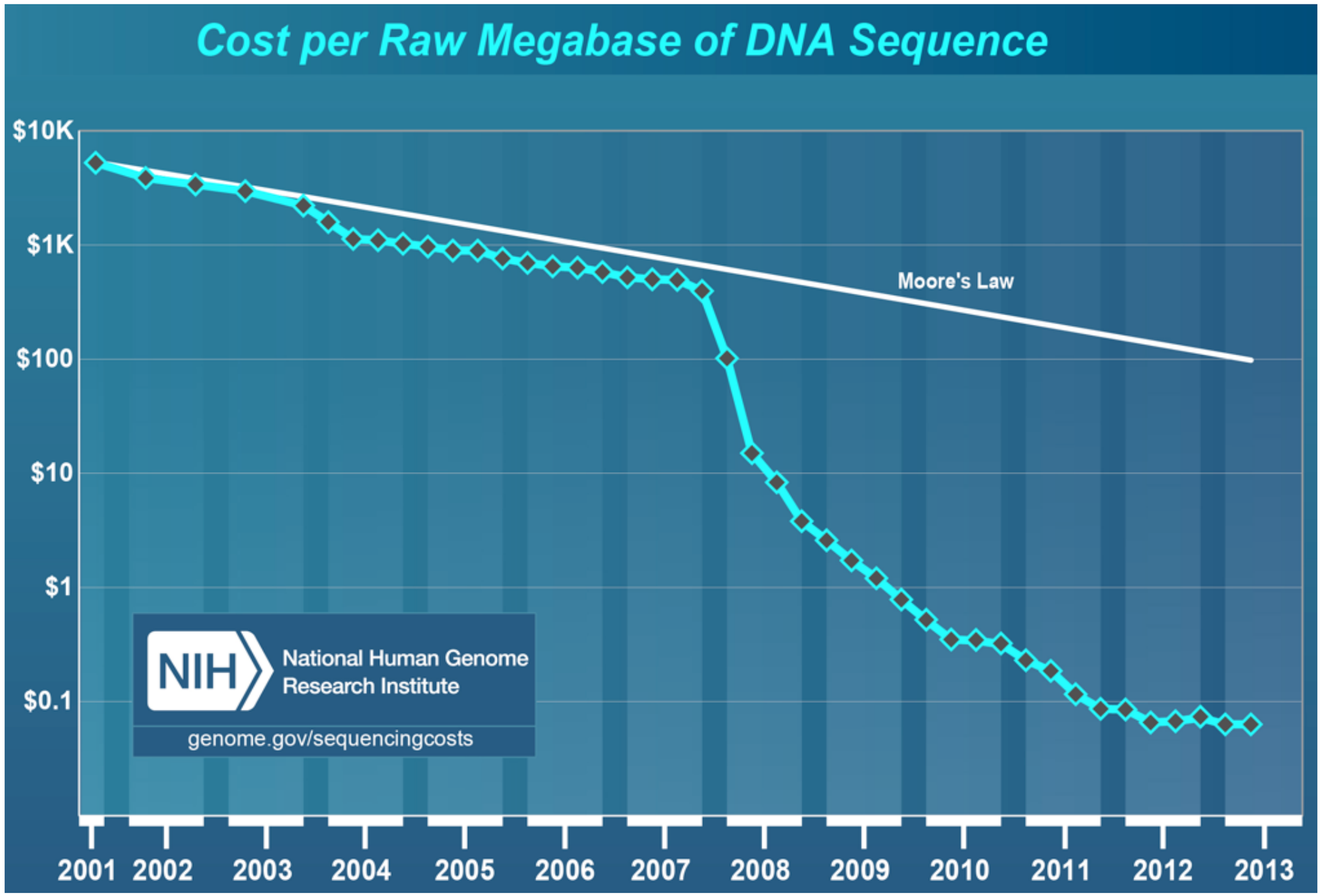
Inputs			Outputs	
A	**B**	**CarryIn**	**CarryOut**	**Sum**
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1

$$\text{Sum} = (a \cdot \overline{b} \cdot \overline{\text{CarryIn}}) + (\overline{a} \cdot b \cdot \overline{\text{CarryIn}}) + (\overline{a} \cdot \overline{b} \cdot \text{CarryIn}) + (a \cdot b \cdot \text{CarryIn})$$

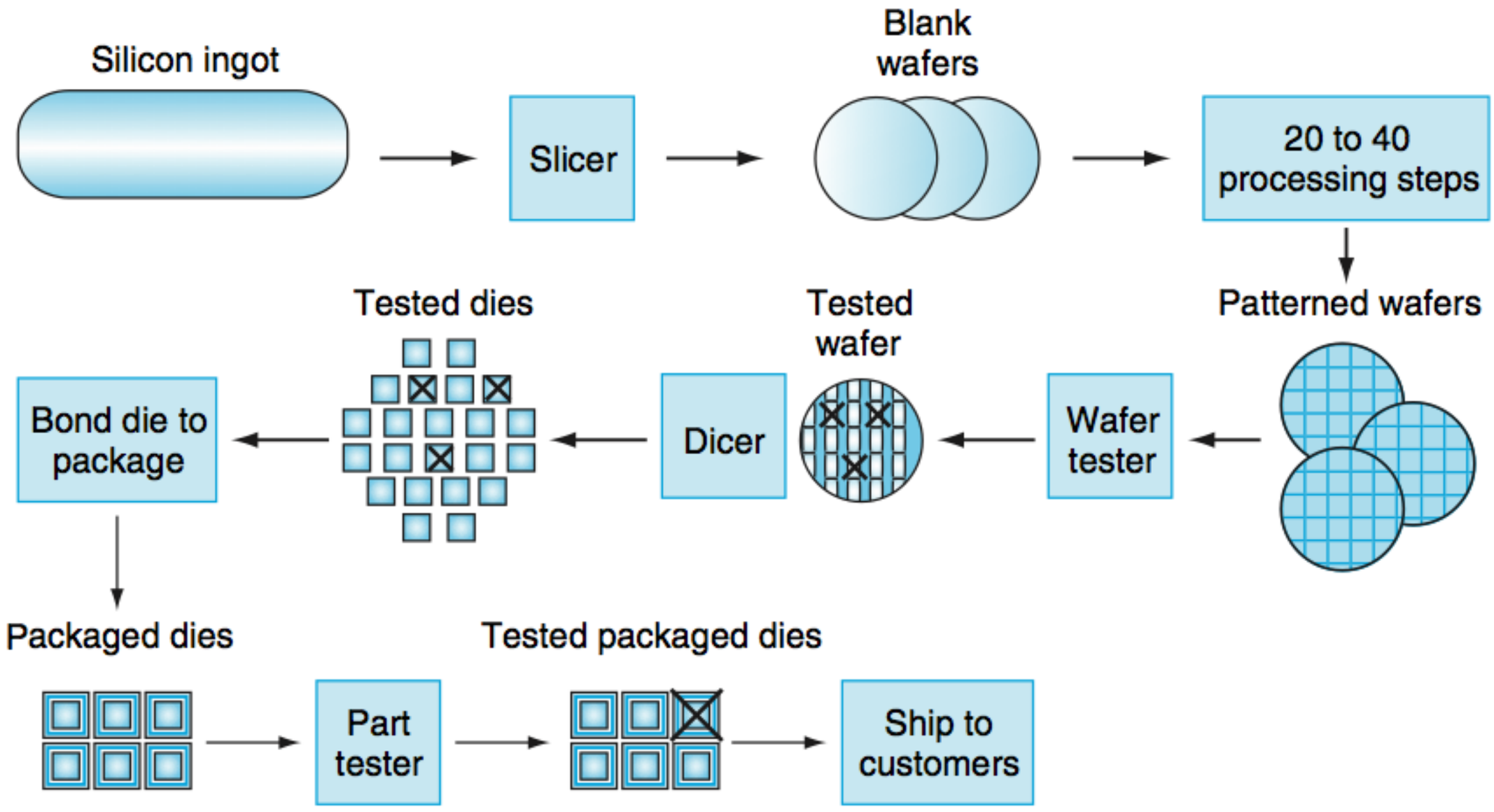
Binary Addition, ctd.

$$
\begin{array}{rll}
 & 0000\ 0000\ 0000\ 0000\ 0000\ 0000\ 0000\ 0111_{two} & = 7_{ten} \\
+ & 0000\ 0000\ 0000\ 0000\ 0000\ 0000\ 0000\ 0110_{two} & = 6_{ten} \\
\hline
= & 0000\ 0000\ 0000\ 0000\ 0000\ 0000\ 0000\ 1101_{two} & = 13_{ten}
\end{array}
$$

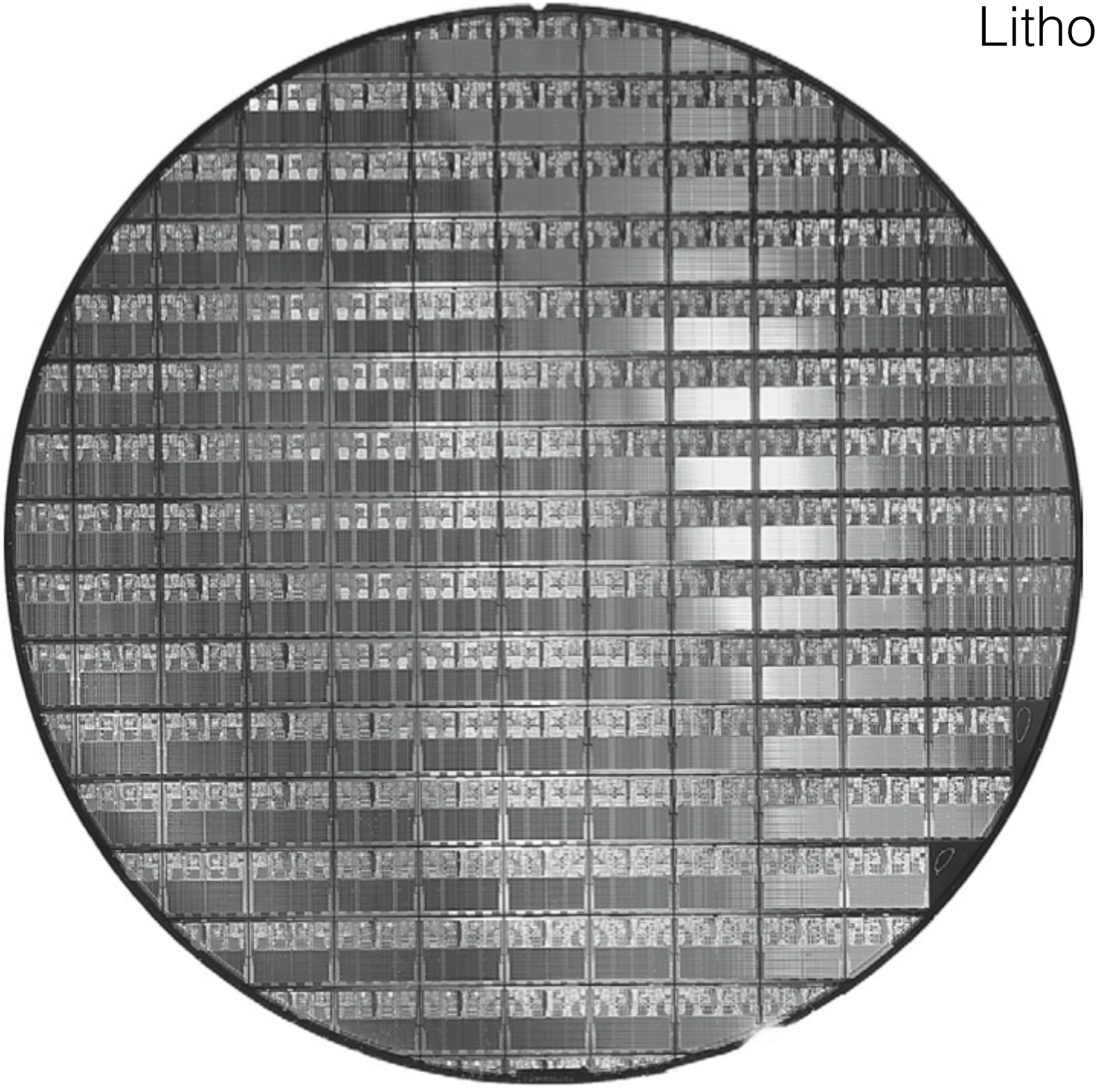


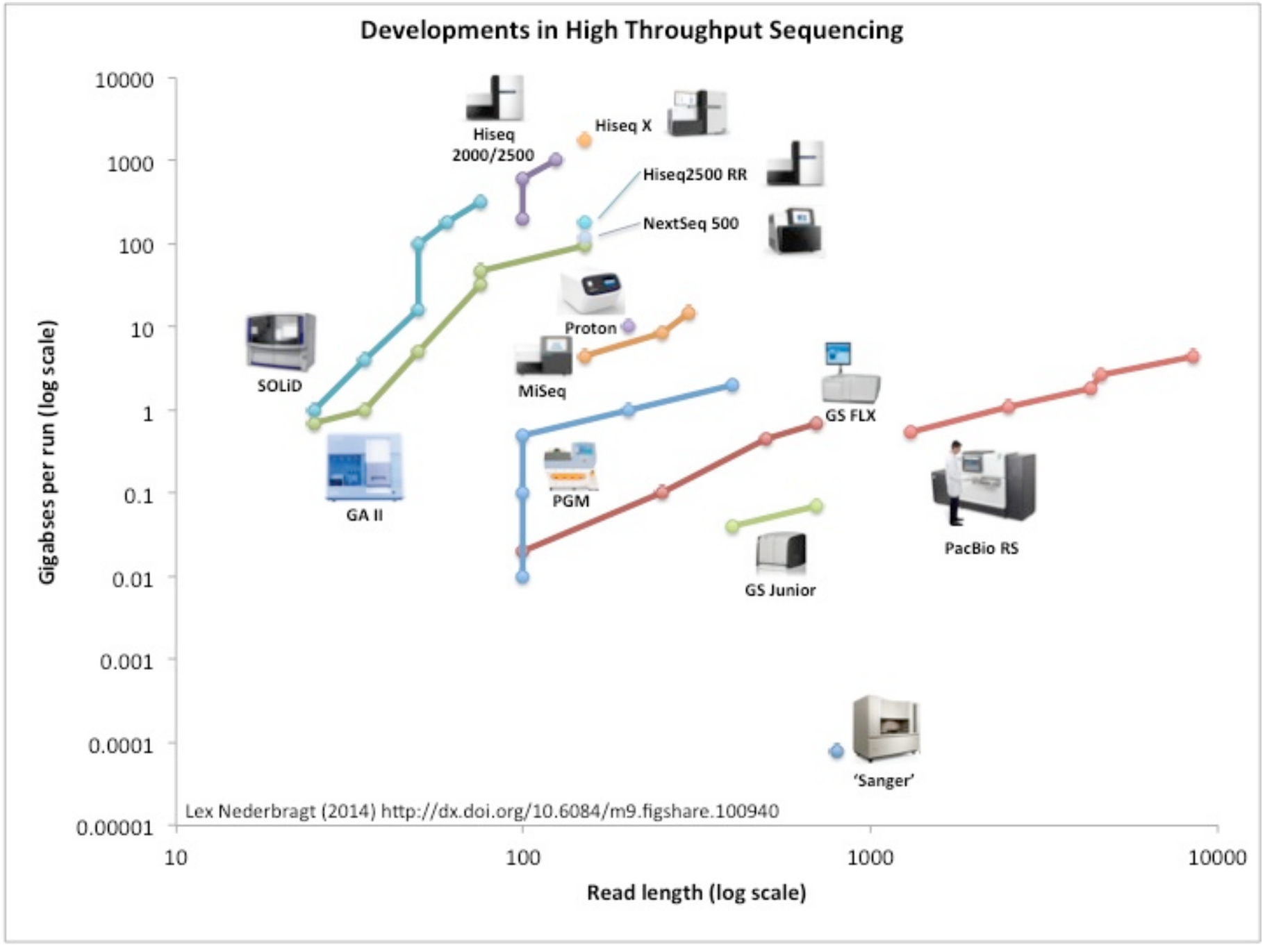


- Some people say it takes 18 months and others say 24
- Some interpret the law to be about the doubling of processing power, not the number of transistors.
- A self-fulfilling prophecy than an actual law, principle or observation: power constraints / parallelism


Silicon ingot
Slicer
Blank wafers
20 to 40 processing steps
Patterned wafers
Wafer tester
Tested wafer
Dicer
Tested dies
Bond die to package
Packaged dies
Part tester
Tested packaged dies
Ship to customers

Lithography




Developments in High Throughput Sequencing
Gigabases per run (log scale)
10000
1000
100
10
1
0.1
0.01
0.001
0.0001
0.00001
Hiseq 2000/2500
Hiseq X
Hiseq2500 RR
NextSeq 500
Proton
SOLiD
MiSeq
GS FLX
GA II
PGM
PacBio RS
GS Junior
'Sanger'
Lex Nederbragt (2014) http://dx.doi.org/10.6084/m9.figshare.100940
10
100
1000
10000
Read length (log scale)