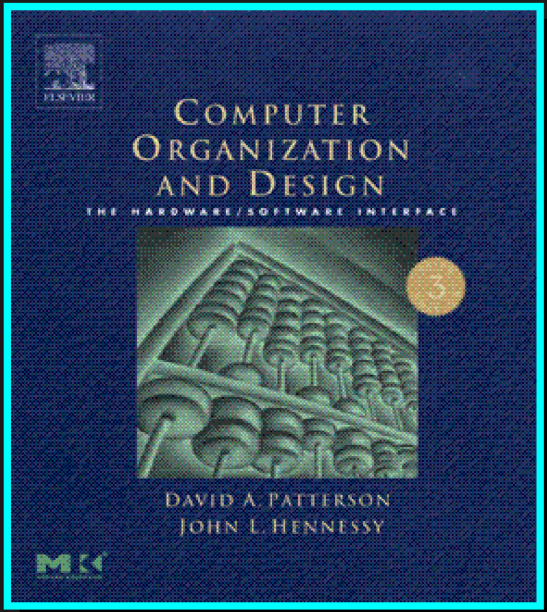
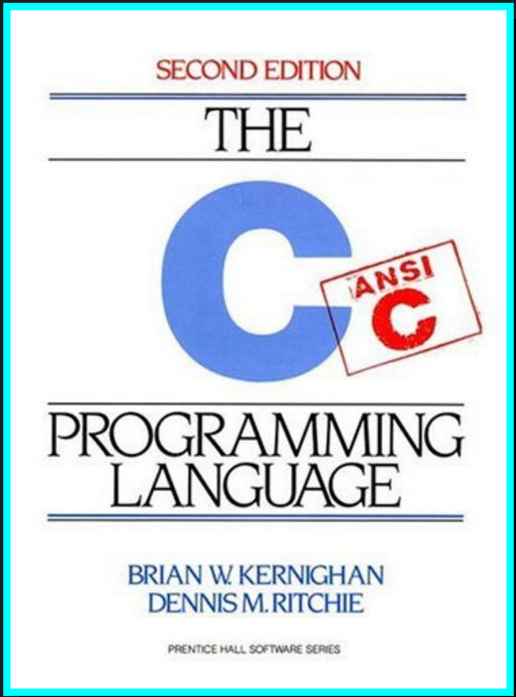
Intro to Math 230
Assembly Language Programming



Lecture # 01

01/15/08



Lecture Overview

- Course Overview
- Short history of industry trends and motivation for course need
- Lab: command line environment review

M230 Course Description

- Hands-on programming course in C and assembly language programming
- Covers low level programming and debugging techniques, computer architecture, input/output programming

Math 230

- Meeting Times
  - Lecture:
    - Tue, Thu: 8:00 am - 9:15 am, rm 394
  - Lab:
    - Tue, Thu: 9:30 am - 10:45 am, rm 394

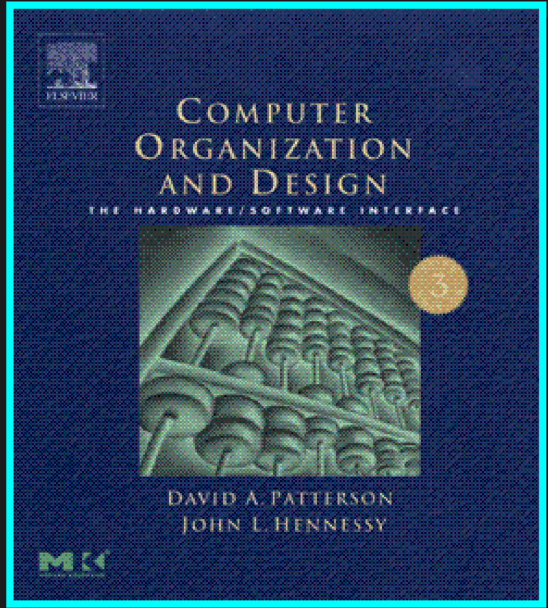
- Class Web Resources:
  - http://swccd.blackboard.com
  - http://groups.google.com/group/swcClassMath230

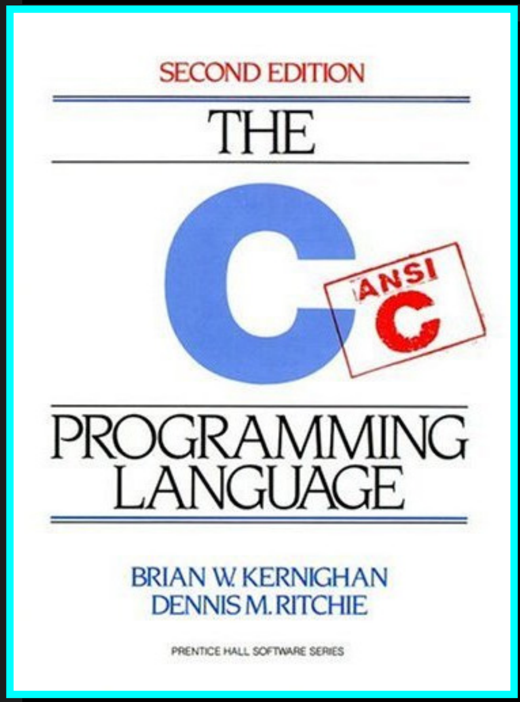
Instructor Contact Info

- Bruce Smith, Assist. Prof of Mathematics
- Phone: 421-6700, x5291
- e-mail: bsmith@swccd.edu
- Office: room 320F
- Office hours:
  - M, W: 12:10 pm - 1:50 pm
  - Fri: 12:00 pm - 12:50 pm
    - You can also contact me to setup an appointment outside these hours!

Textbook and Materials

- Required:
  - *Computer Organization and Design, 3rd Edition*, by Patterson and Hennessy
  - *Programming in C, 2nd Edition*, by Kernighan and Ritchie (K&R)
- On library reserve:
  - *Digital Principles and Applications*, by Leach
  - K&R





Attendance

- You can be dropped if you have more than 4 absences (i.e., 2 weeks worth of classes)
- Tardiness and early departures may also be counted as absences
- Course material discussed during the Lab section may extend (or even add to) lecture material. You are responsible for all material covered in both lab and lecture

Evaluation Policy

- Semester Grade

Evaluation Policy		
Labs	(~10)	10%
Homework	(~10)	10%
Projects	(~5)	30%
Midterm Exams	(2)	25%
Final Exam		25%
Total:		**100%**

Assigned reading: *P&H*, Ch 1, 3.1, 3.2 (exclude Fig.3.1)
HW01 (due Thu, 1/24): Exercises 1.1 thru 1.28, 1.54
Lab01 (due Thu, 1/24): see Blackboard

Homework, Labs and Projects

- Lab exercises
  - every week; to be submitted by the Thursday lab session

- Homework exercises
  - ~every week
- Projects
  - ~every 3 weeks

Class Policies

- No food or drink (water bottles OK)
- Cell phones silent
- No children or visitors without prior permission

Class Policies

- If you are found cheating or helping someone cheat, you may receive as much as –50% of the assignment's value

- Any further cheating will result in expulsion from the course.
  - Also see SWC Course Catalog regarding student conduct.

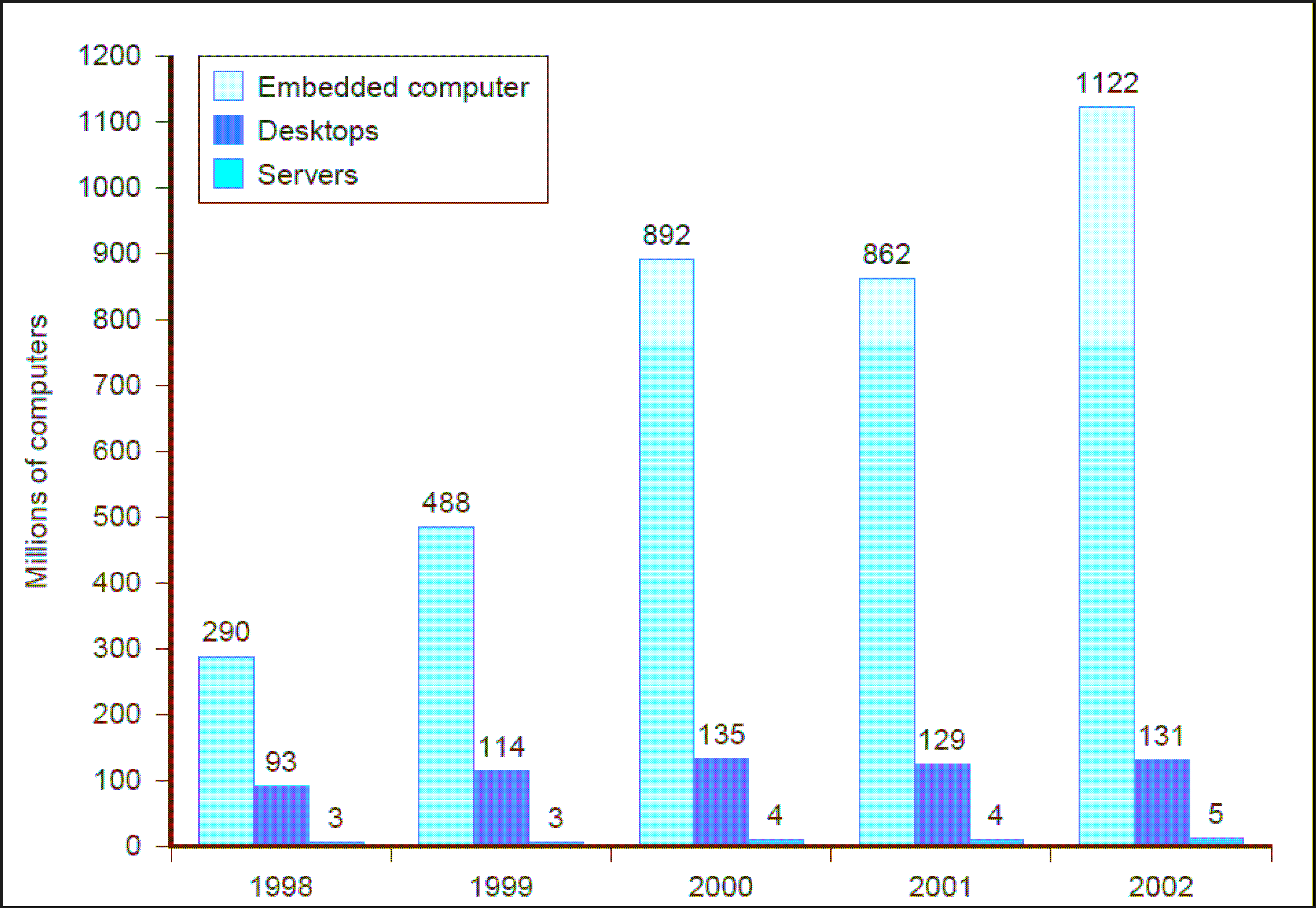
Class Resources

- Primarily the info that is packaged with your textbook
  - PCspim
    - compiler/assembler
    - see text book
  - MARs
  - Science of Computing Text
  - Camera
  - Vivio
  - LogiSim

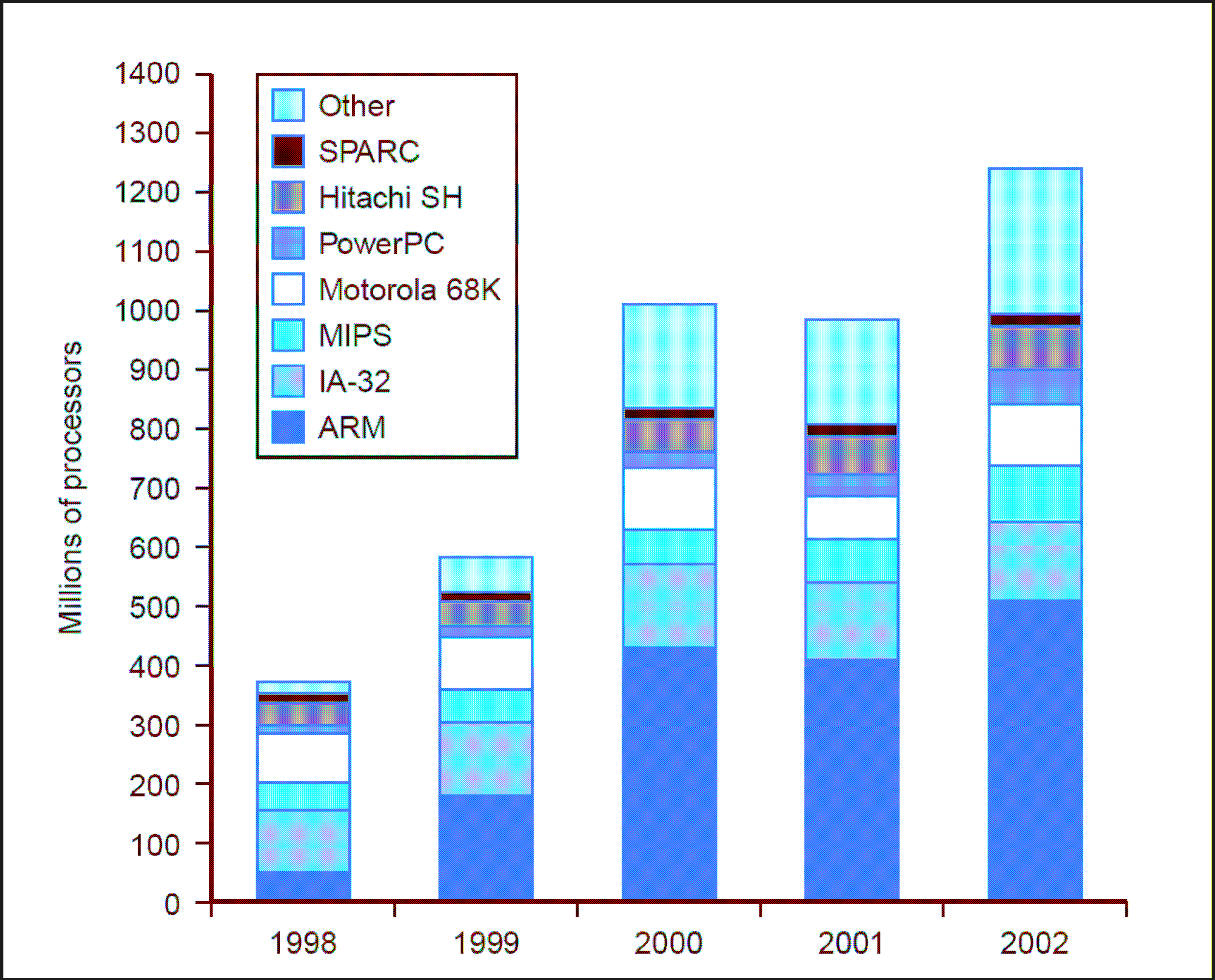
Is MIPS relevant?

- MIPS
  - **Microprocessor Without Interlocked Pipeline Stages**

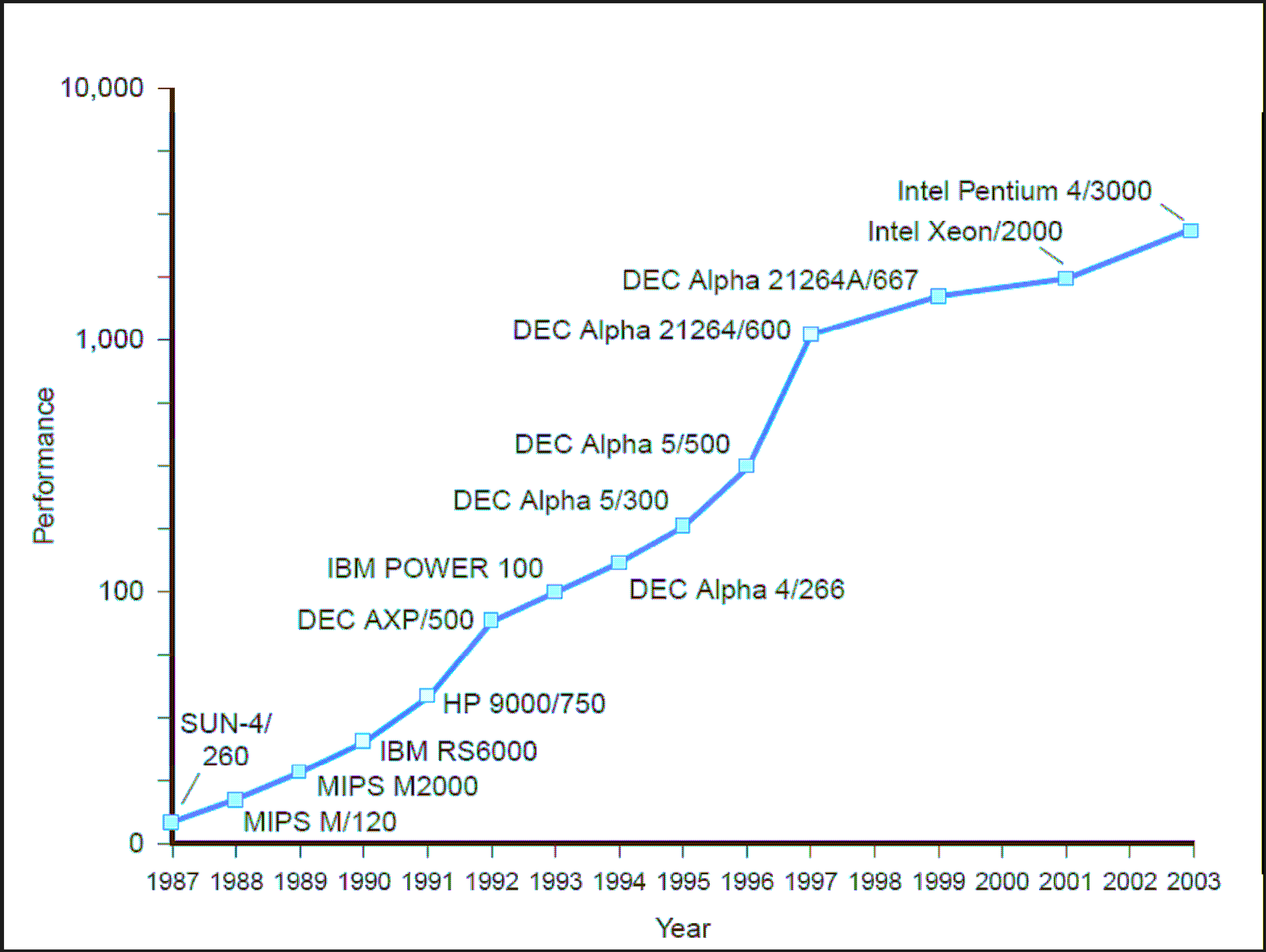
Market Share



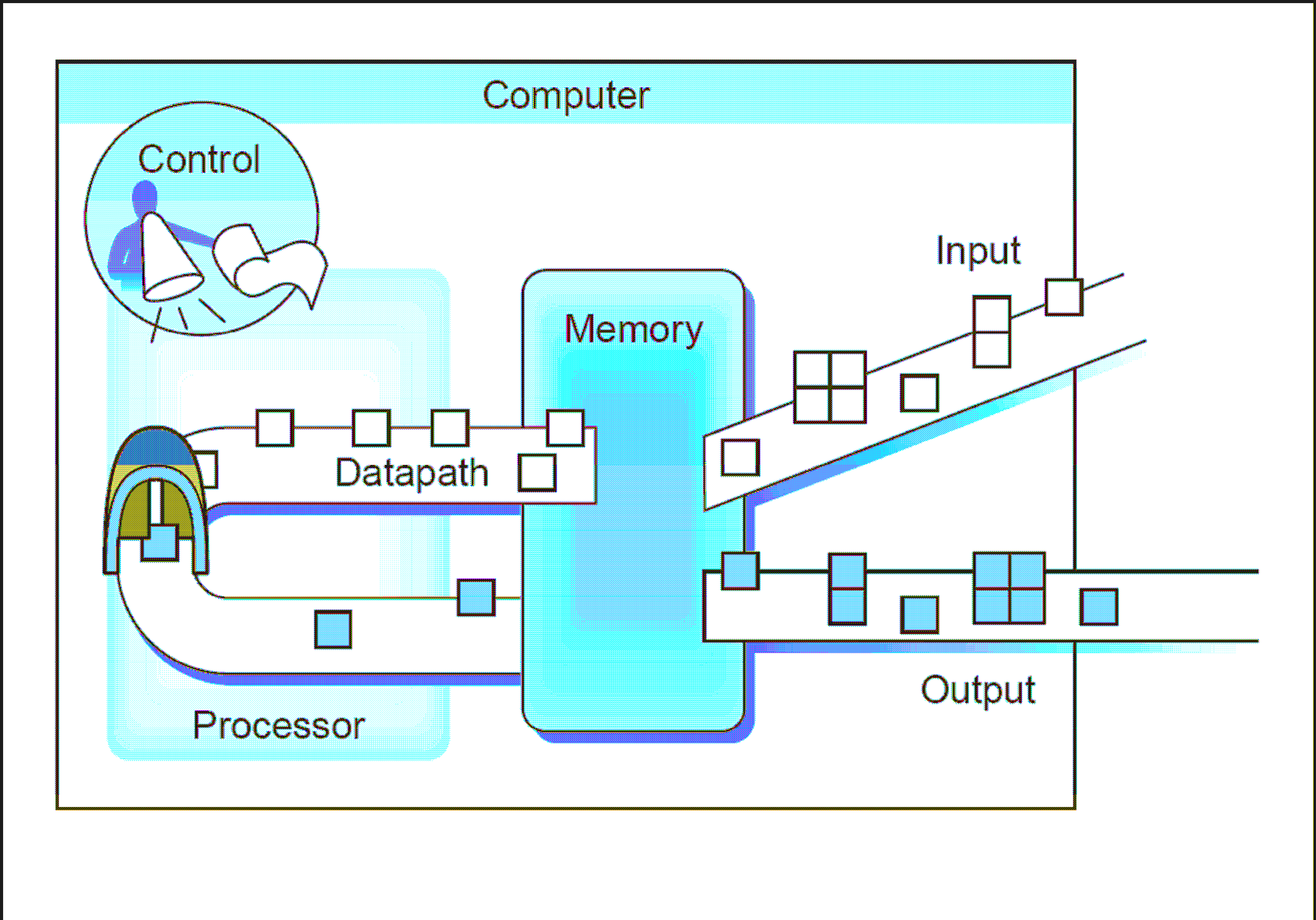
The MIPS share



Workstation Peformance



Framework for the Course



Assembly Language

Assembly Language

- Machine language

Assembly Language

- Machine language
  - The numeric language understood by a computer's processor (the CPU).

Assembly Language

- Machine language
  - The numeric language understood by a computer's processor (the CPU).
  - Consists entirely of 1's and 0's
    - low and high voltages

Assembly Language

- Machine language
  - The numeric language understood by a computer's processor (the CPU).
  - Consists entirely of 1's and 0's
    - low and high voltages
  - The 1's and 0's are usually grouped and represented as larger numbers (hex or octal)

Assembly Language

- Machine language
  - The numeric language understood by a computer's processor (the CPU).
  - Consists entirely of 1's and 0's
    - low and high voltages
  - The 1's and 0's are usually grouped and represented as larger numbers (hex or octal)
- Assembly Language

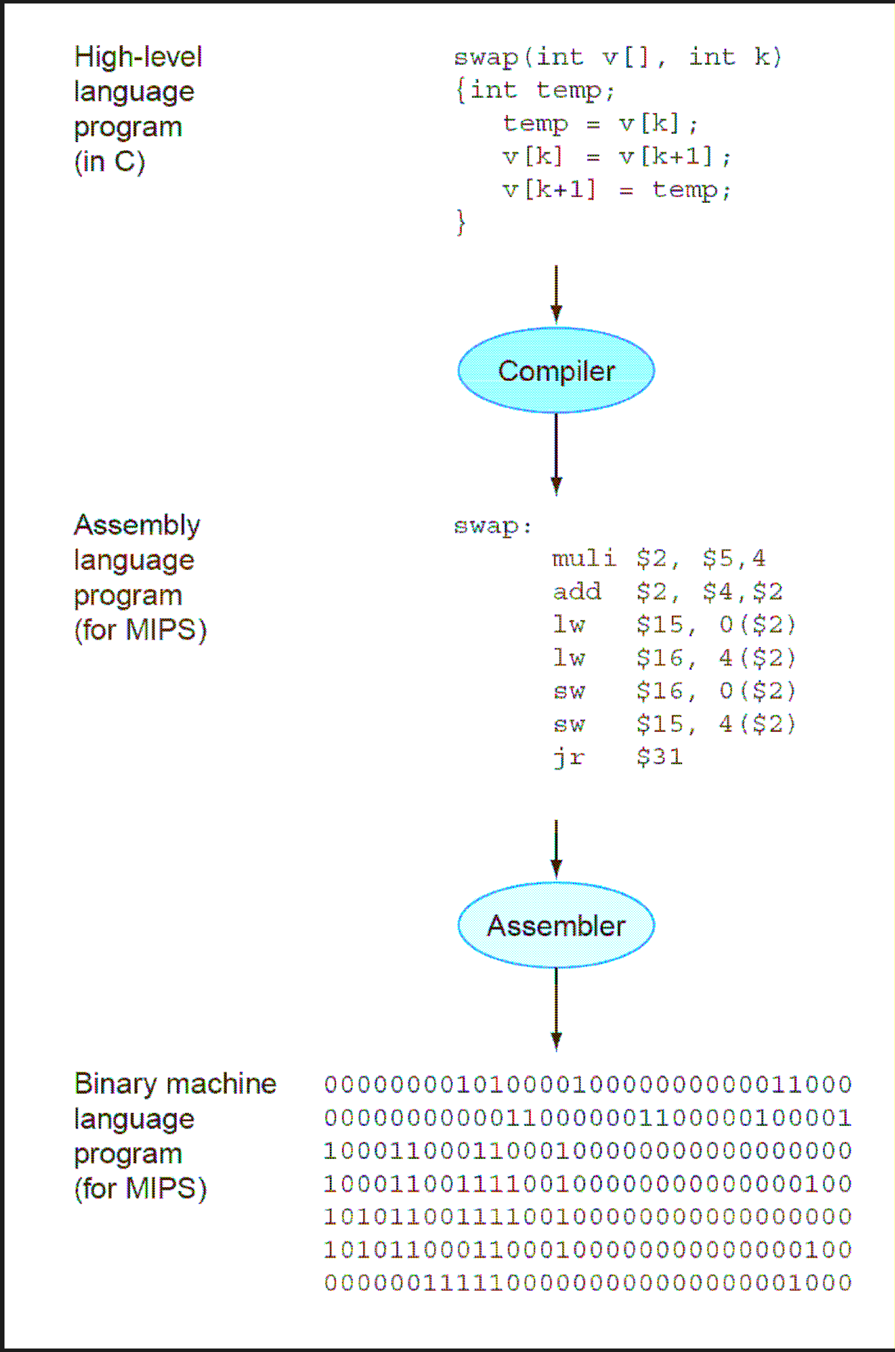
Assembly Language

- Machine language
  - The numeric language understood by a computer's processor (the CPU).
  - Consists entirely of 1's and 0's
    - low and high voltages
  - The 1's and 0's are usually grouped and represented as larger numbers (hex or octal)
- Assembly Language
  - Short mnemonics to represent the numeric (machine) language, e.g., ADD, LW, SUB, MUL, J, DIV, JAL

Assembly Language

- Machine language
  - The numeric language understood by a computer's processor (the CPU).
  - Consists entirely of 1's and 0's
    - low and high voltages
  - The 1's and 0's are usually grouped and represented as larger numbers (hex or octal)
- Assembly Language
  - Short mnemonics to represent the numeric (machine) language, e.g., ADD, LW, SUB, MUL, J, DIV, JAL
  - Converted to machine language

Babel

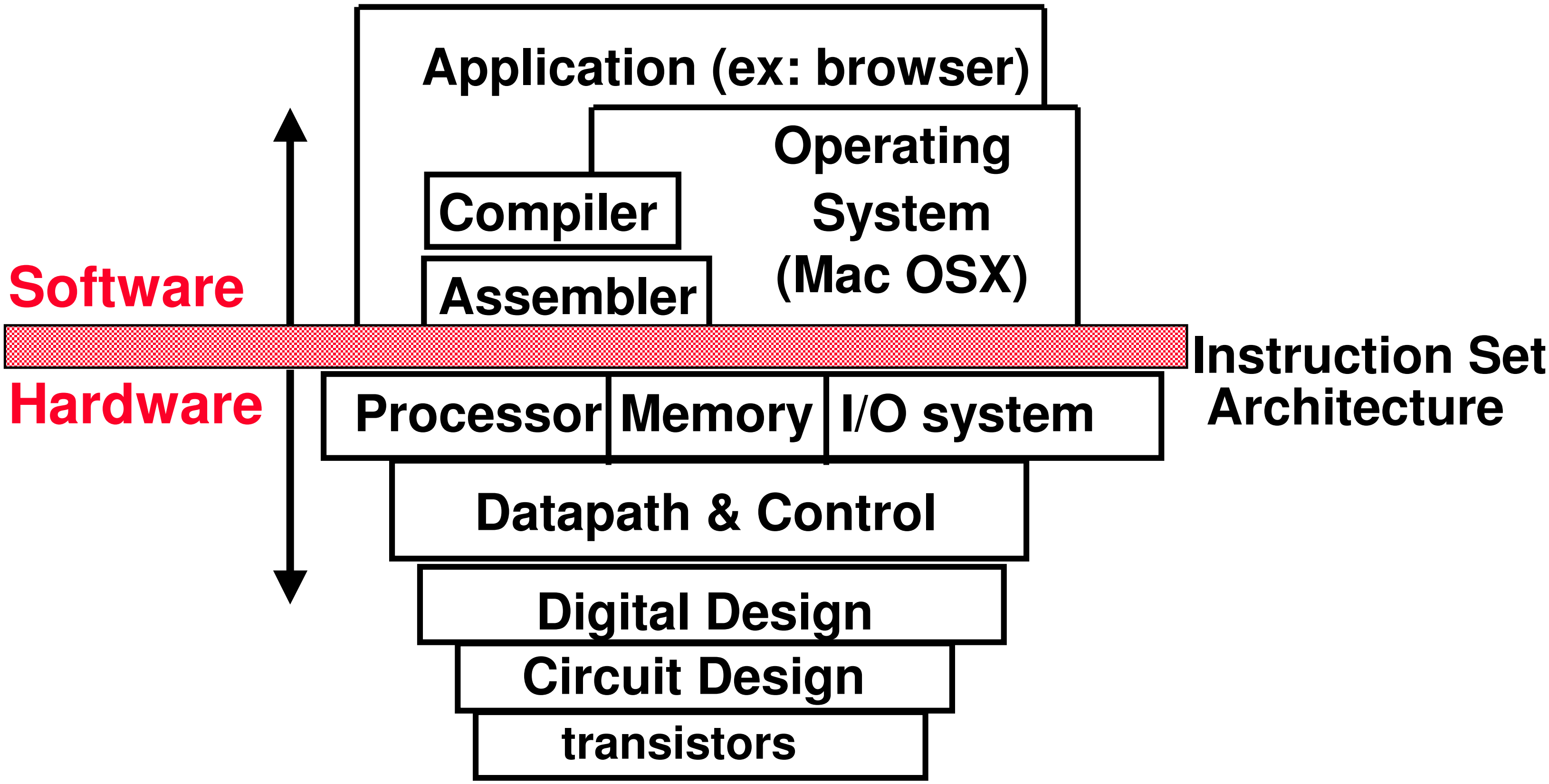


High-level language program (in C)
swap(int v[], int k)
{int temp;
temp = v[k];
v[k] = v[k+1];
v[k+1] = temp;
}
Compiler
Assembly language program (for MIPS)
swap:
muli $2, $5,4
add $2, $4,$2
lw $15, 0($2)
lw $16, 4($2)
sw $16, 0($2)
sw $15, 4($2)
jr $31
Assembler
Binary machine language program (for MIPS)
00000000101000010000000000011000
00000000000110000001100000100001
10001100011000100000000000000000
10001100111100100000000000000100
10101100111100100000000000000000
10101100011000100000000000000100
00000011111000000000000000001000

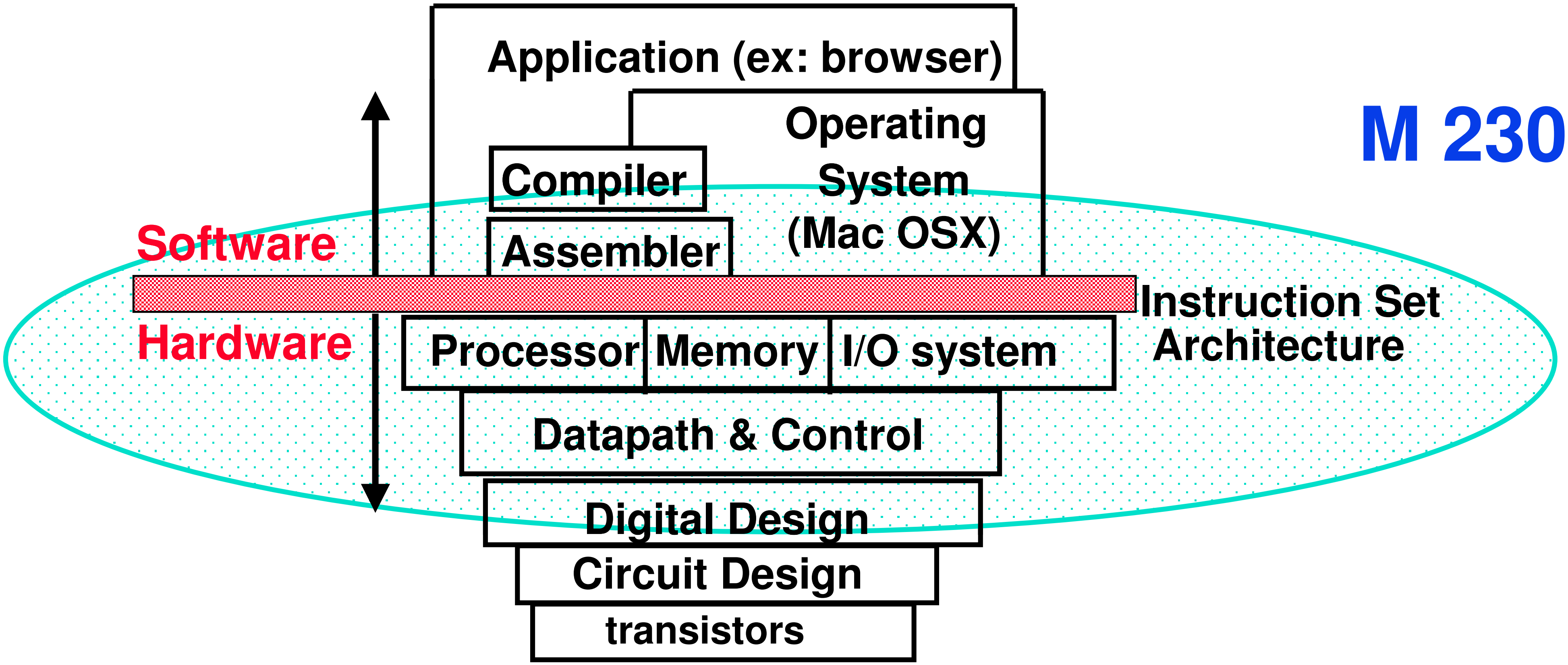
Applications

- Embedded systems
  - flight control
  - air-conditioning
  - home alarm
  - digital phone
- Create device drivers
  - printer drivers
  - USB drivers
- Computer games needing HW access
  - small, quick code

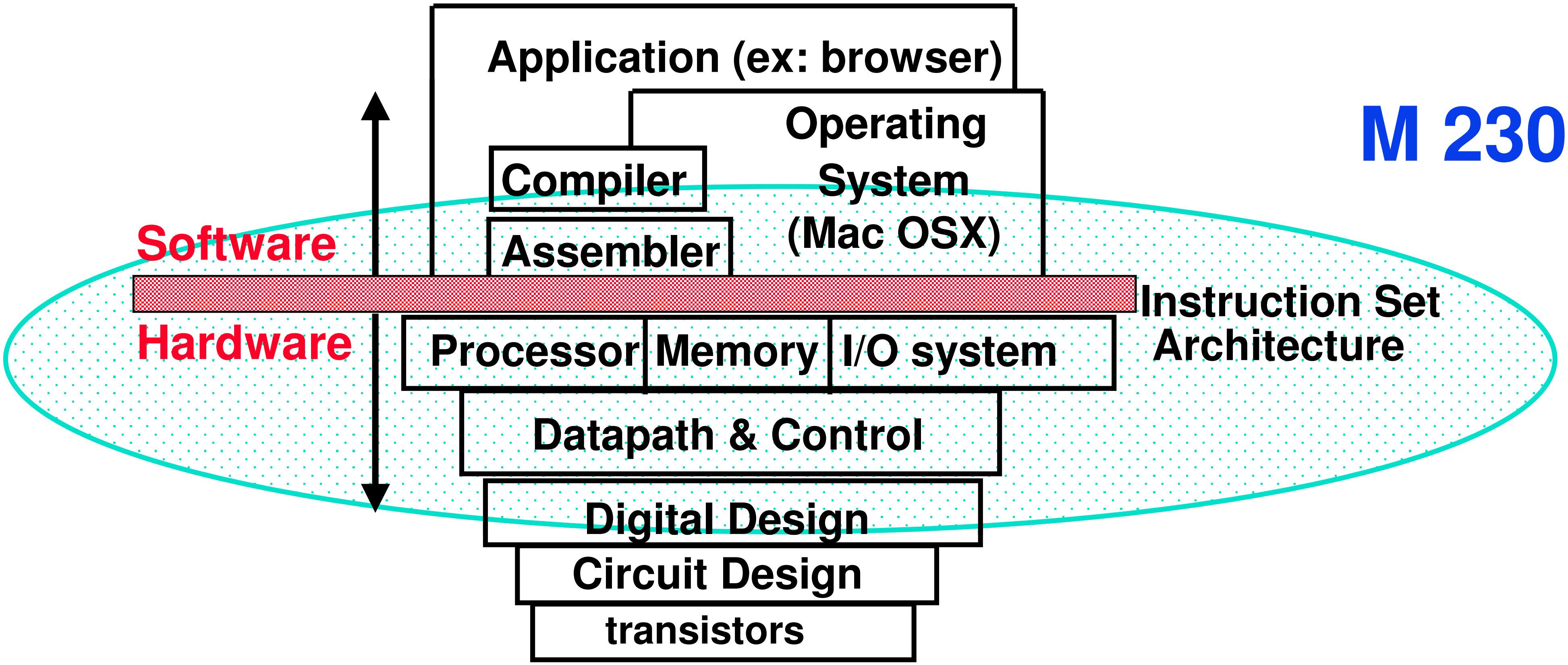
What are "Machine Structures"?



What are "Machine Structures"?

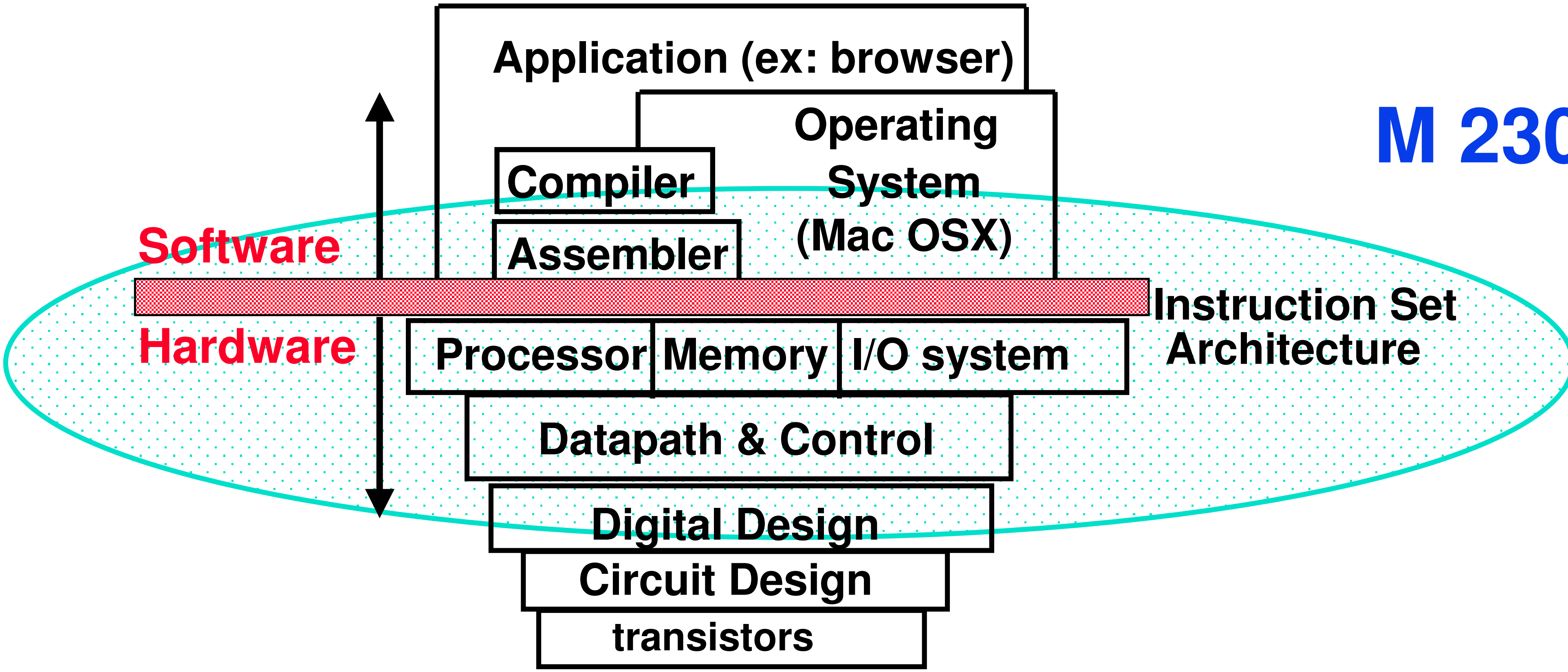


What are "Machine Structures"?



* Coordination of many

What are "Machine Structures"?



M 230

* Coordination of many

levels (layers) of abstraction

Levels of Representation

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

Compiler

Assembly Language Program (e.g.,MIPS)

Assembler

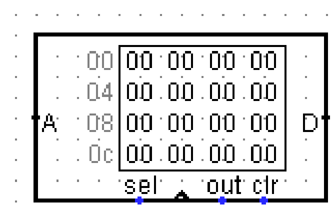
Machine Language Program (MIPS)

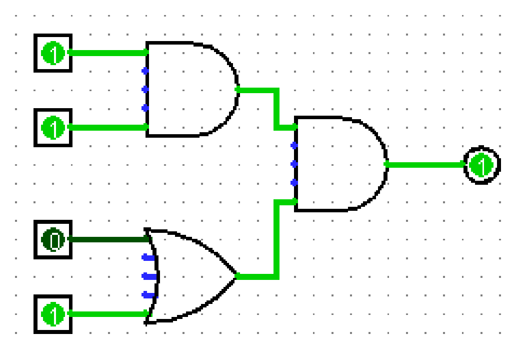
Machine Interpretation

Hardware Architecture Description (Logic, Logisim, Verilog, etc.)

Architecture Implementation

Logic Circuit Description (Logisim, etc.)





Levels of Representation

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

Compiler

Assembly Language Program (e.g.,MIPS)

Assembler

Machine Language Program (MIPS)

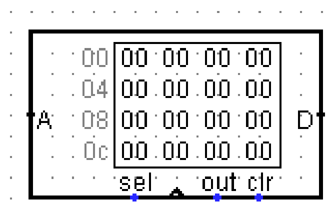
Machine Interpretation

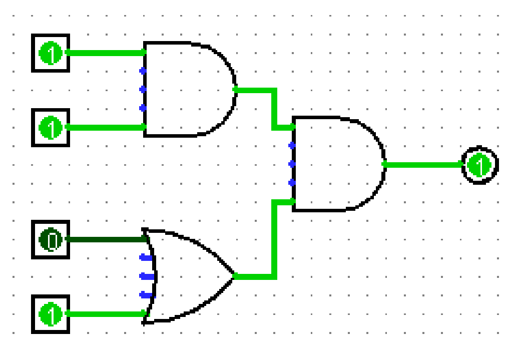
Hardware Architecture Description (Logic, Logisim, Verilog, etc.)

Architecture Implementation

Logic Circuit Description (Logisim, etc.)

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





Levels of Representation

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

Compiler

Assembly Language Program (e.g.,MIPS)

Assembler

Machine Language Program (MIPS)

Machine Interpretation

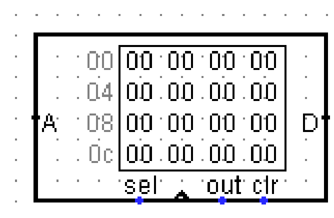
Hardware Architecture Description (Logic, Logisim, Verilog, etc.)

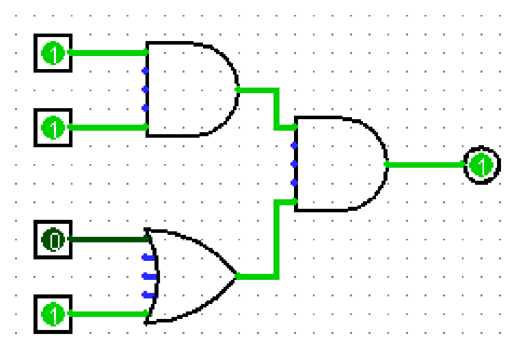
Architecture Implementation

Logic Circuit Description (Logisim, etc.)

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

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





Levels of Representation

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

Compiler

Assembly Language Program (e.g.,MIPS)

Assembler

Machine Language Program (MIPS)

Machine Interpretation

Hardware Architecture Description (Logic, Logisim, Verilog, etc.)

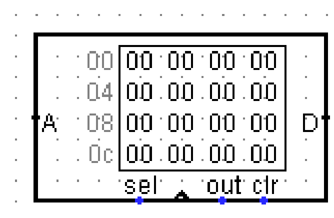
Architecture Implementation

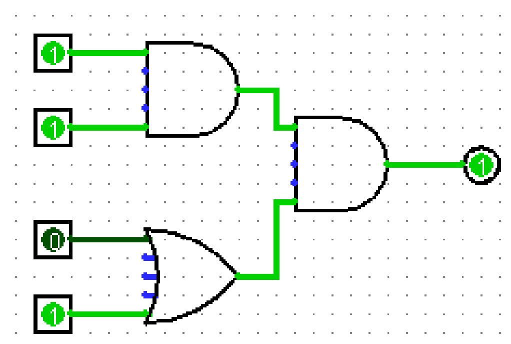
Logic Circuit Description (Logisim, etc.)

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

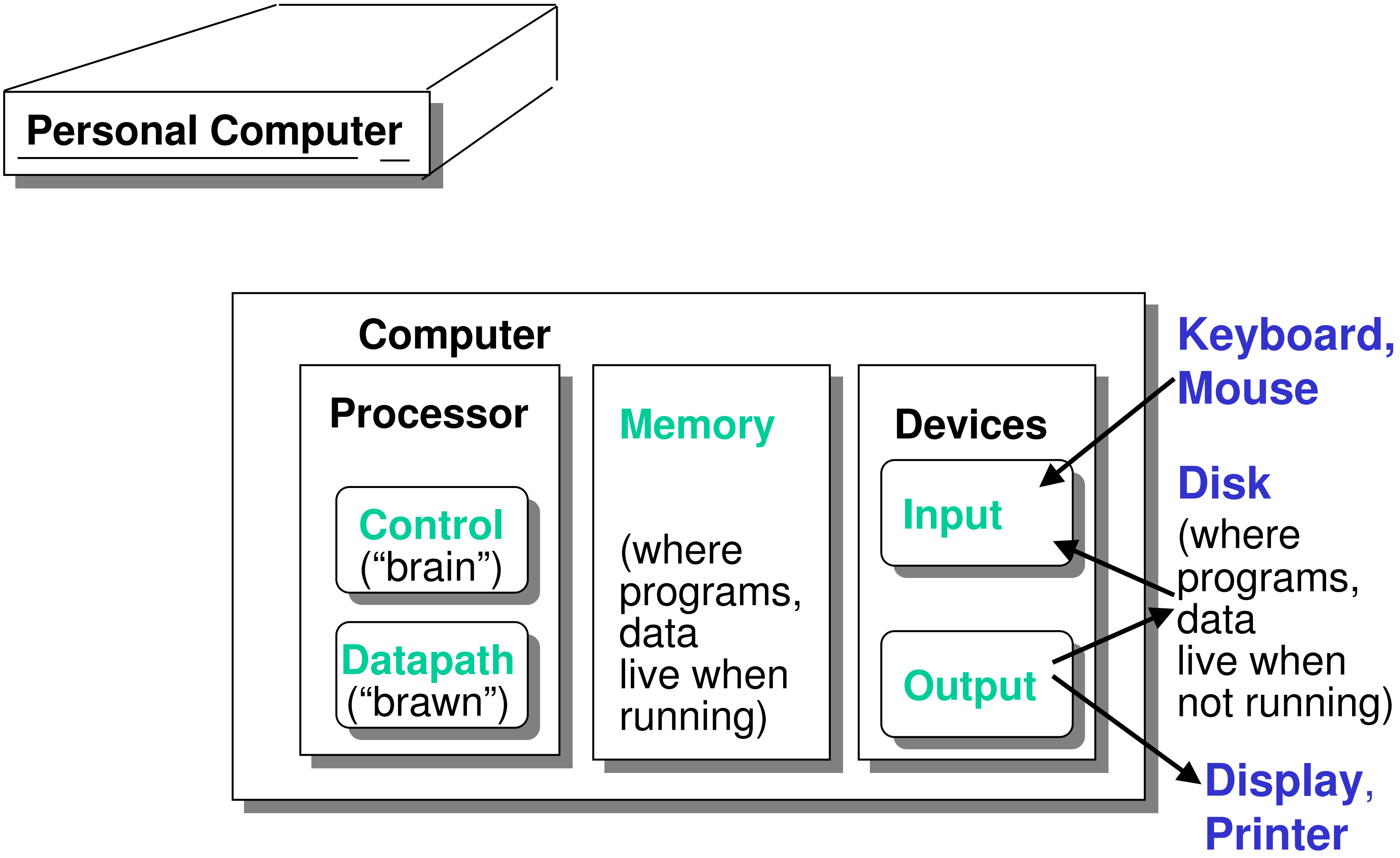
```
lw   $t0, 0($2)
lw   $t1, 4($2)
sw   $t1, 0($2)
sw   $t0, 4($2)
```

```
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
```





Anatomy: 5 components of any Computer



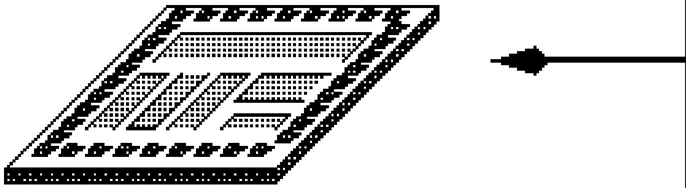
Overview of Physical Implementations

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.

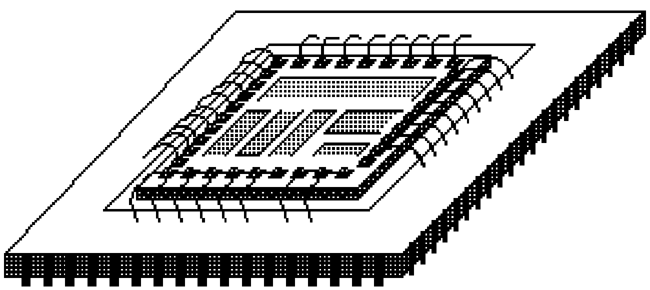
Integrated Circuits (2005 state-of-the-art)

Bare Die



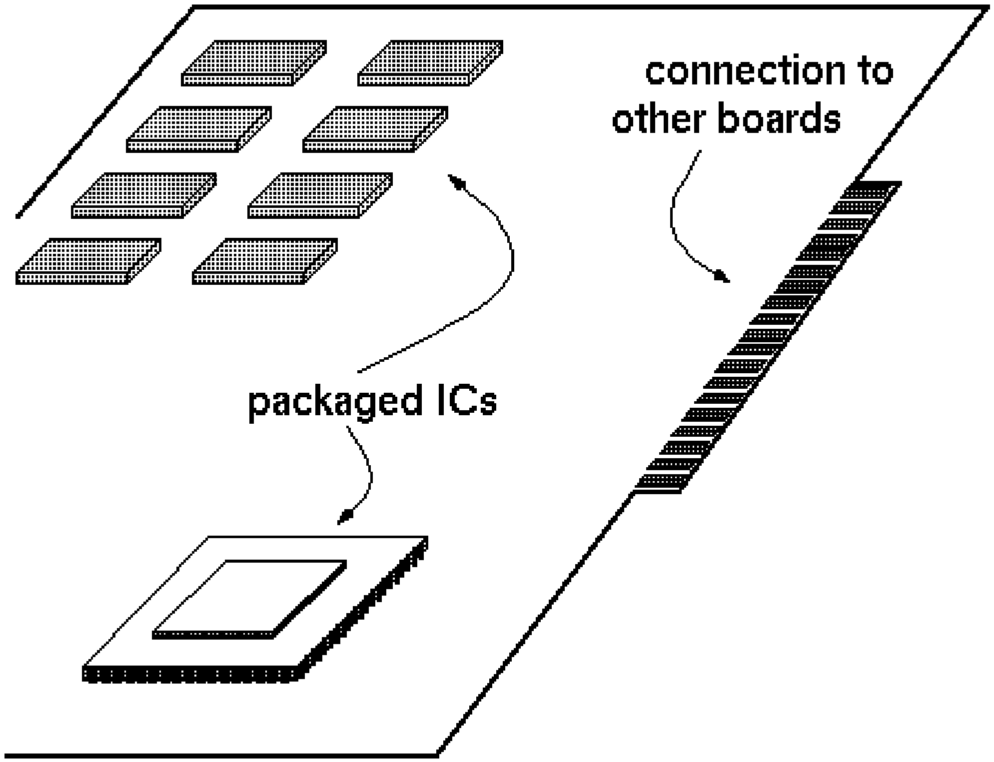
- Primarily Crystalline Silicon
- 1mm - 25mm on a side
- 2005 - feature size ~ 90 nm = 90 x 10^{-9}m
- 100 - 1000M transistors
- (25 - 100M "logic gates")
- 3 - 10 conductive layers
- "CMOS" (complementary metal oxide semiconductor) - most common.

Chip in Package



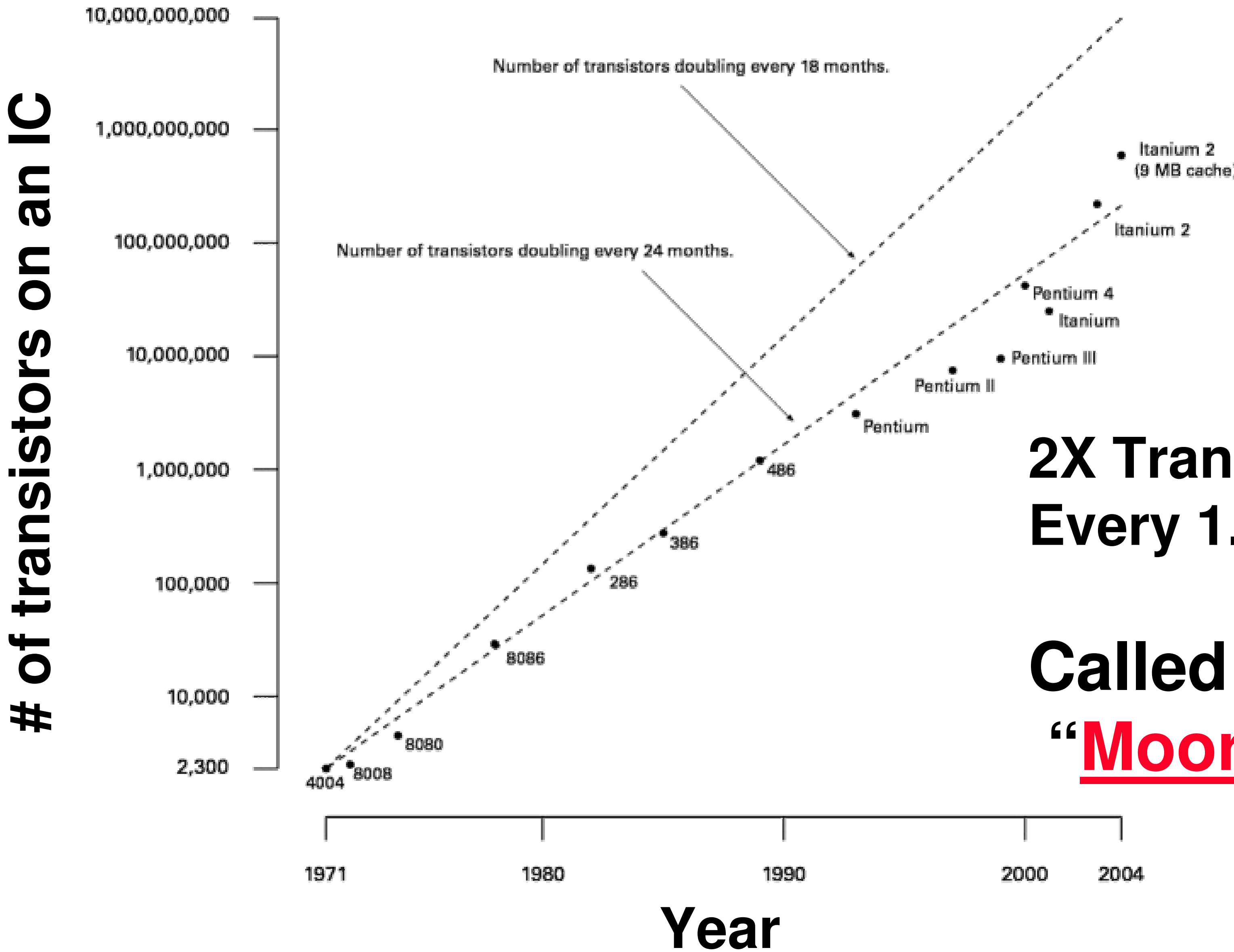
- Package provides:
  - spreading of chip-level signal paths to board-level
  - heat dissipation.
- Ceramic or plastic with gold wires.

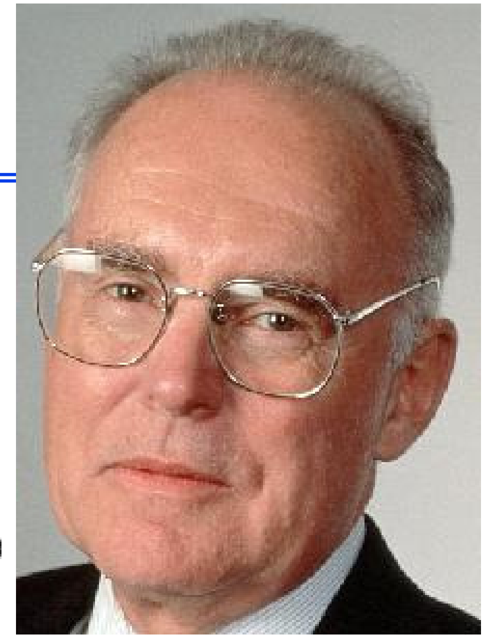
Printed Circuit Boards



- fiberglass or ceramic
- 1-20 conductive layers
- 1-20 in on a side
- IC packages are soldered down.
- Provides:
  - Mechanical support
  - Distribution of power and heat.

Technology Trends: Microprocessor Complexity





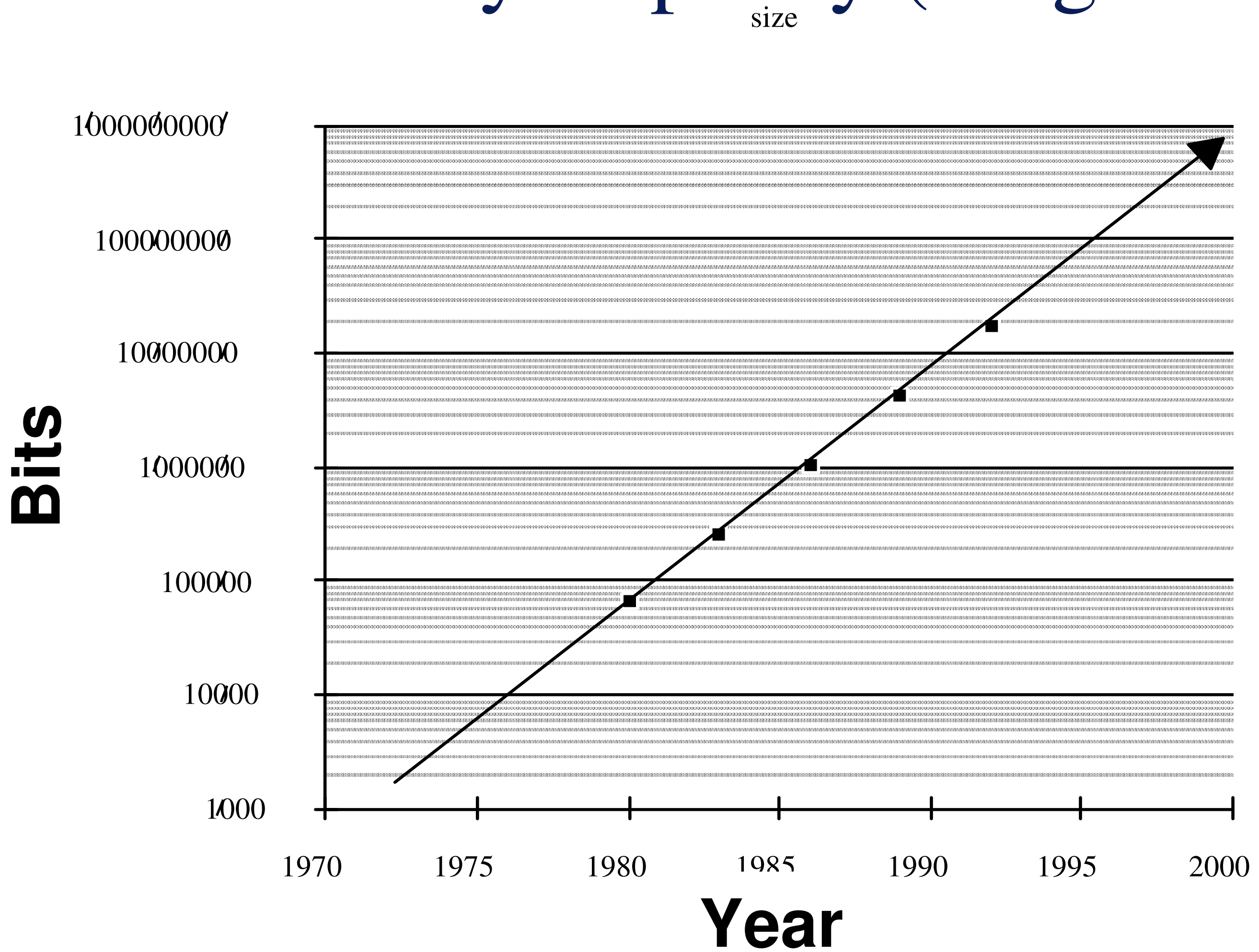
Gordon Moore
Intel Cofounder

2X Transistors / Chip Every 1.5 years

Called "Moore's Law"

Technology Trends:

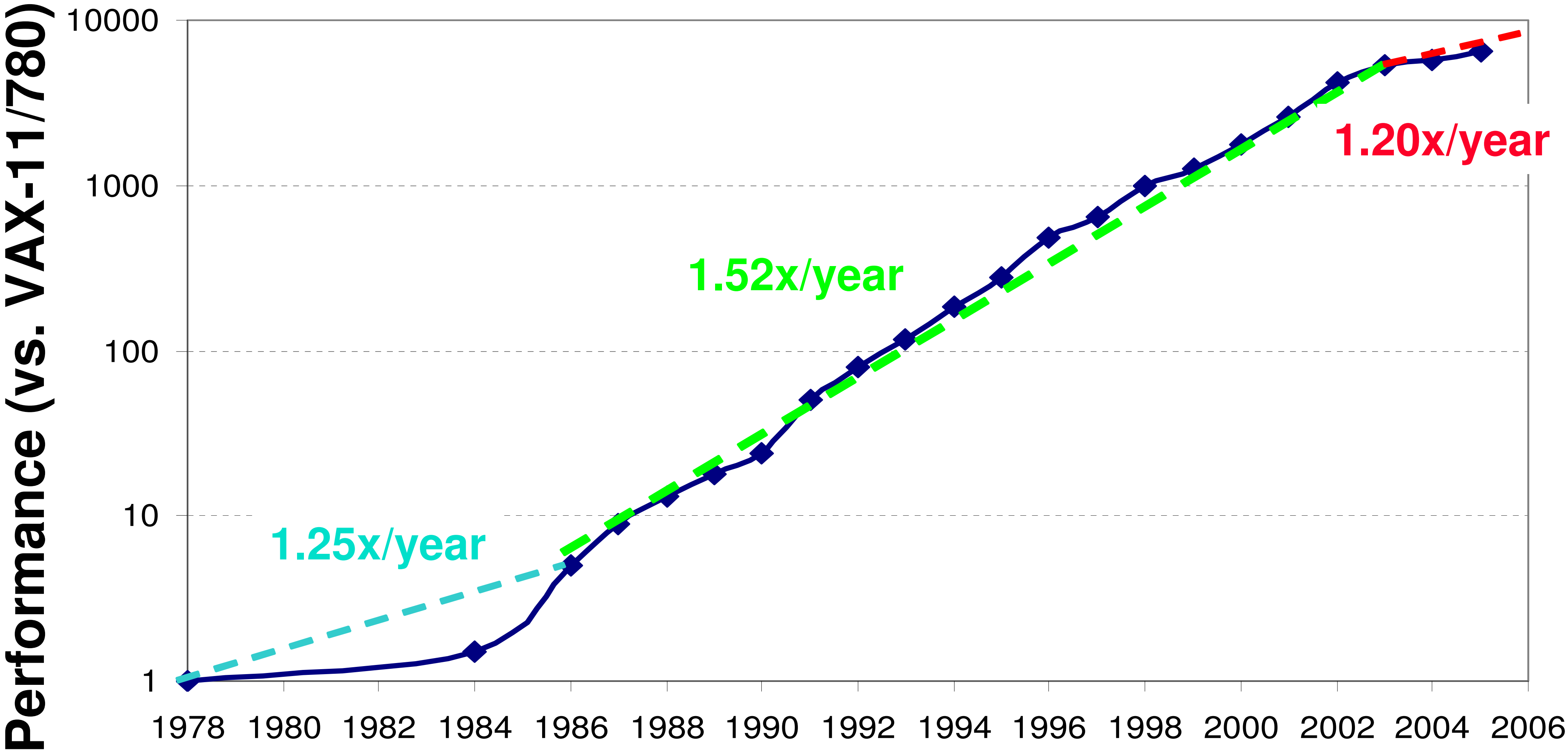
Memory Capacity (Single-Chip DRAM)



year	size (Mbit)
1980	0.0625
1983	0.25
1986	1
1989	4
1992	16
1996	64
1998	128
2000	256
2002	512
2004	1024 (1Gbit)

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

Technology Trends: Uniprocessor Performance (SPECint)



- VAX : 1.25x/year 1978 to 1986
- RISC + x86: 1.52x/year 1986 to 2002
- RISC + x86: 1.20x/year 2002 to present

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); [slowing!] 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: 8000 MegaBytes (8.0 GigaBytes)
  - Disk capacity: 2000 GigaBytes (2.0 TeraBytes)
  - New units! Mega ⇒ Giga, Giga ⇒ Tera

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

M230: So what's in it for me?

- Learn some of the big ideas in CS & engineering:
  - Principle of abstraction, used to build systems as layers
  - 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
  - Compilation v. interpretation thru system layers
  - Principles/Pitfalls of Performance Measurement

Others Skills learned in 230

- 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'll learn just the basics of hardware design

Course Lecture Outline

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

Today's Lab Work

- Log onto computer (swcstudent):
  - username: pmath
  - password: 7pmath
- Log onto Blackboard
  - swccd.edu or swccd.blackboard.com
    - announcements
    - grades
    - Updating E-mail on Webadvisor
    - sending e-mail
- Explore command-line environment

Lab 0

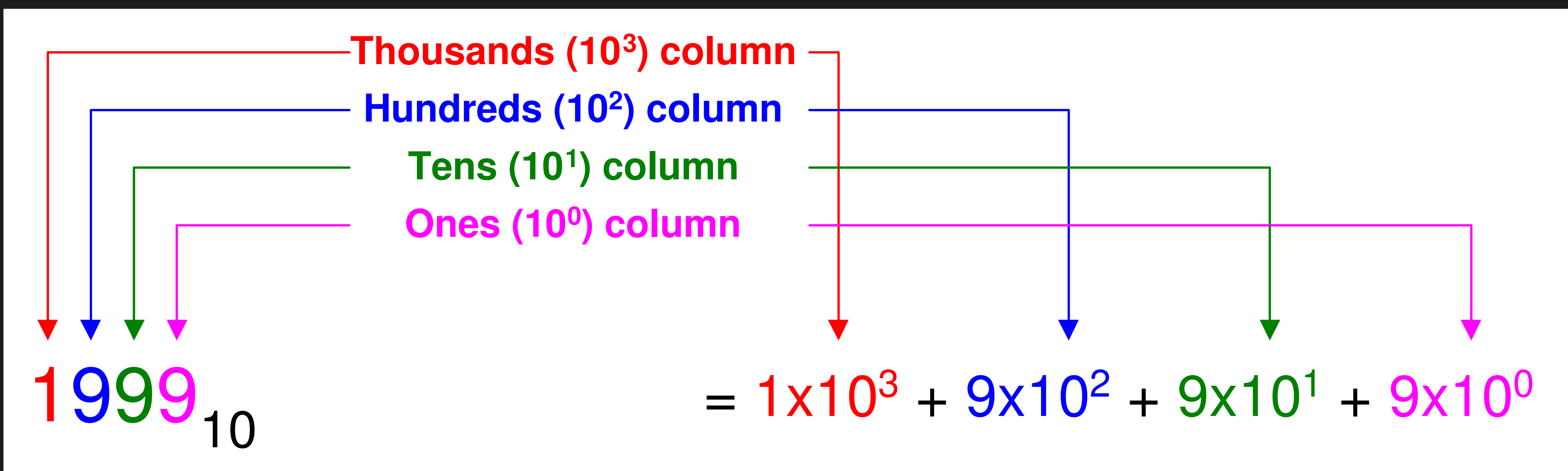
- dir
- cd
- del
- pwd
- pushd/popd
- more
- path
- set
- redirection
- wildcards

Binary Numbers

- Digits are 1 and 0
  - 1 = true
  - 0 = false
- MSB – most significant bit
- LSB – least significant bit
- Bit numbering:

Base-10 (decimal) arithmetic

- Uses the ***ten*** numbers from 0 to 9
- Each column represents a power of ***10***

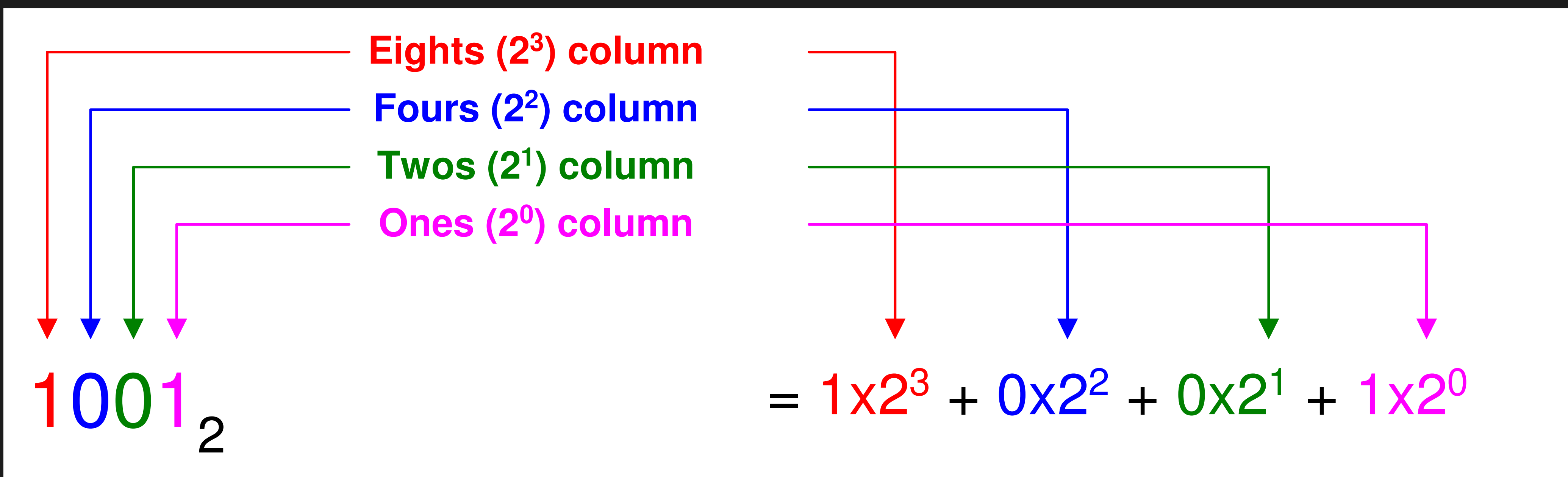


Base-2 (binary) arithmetic

- Uses the ***two*** numbers from 0 to 1
- Every column represents a power of ***2***

Base-2 (binary) arithmetic

- Uses the ***two*** numbers from 0 to 1
- Every column represents a power of ***2***



Translating Binary to Decimal

Weighted positional notation shows how to calculate the decimal value of each binary bit:

$$dec = (D_{n-1} \times 2^{n-1}) + (D_{n-2} \times 2^{n-2}) + ... + (D_1 \times 2^1) + (D_0 \times 2^0)$$

D = binary digit

binary 00001001 = decimal 9:

$$(1 \times 2^3) + (1 \times 2^0) = 9$$

Converting from base-2 to base-10

32	*16*	*8*	*4*	*2*	*1*	
		1	0	0	1	=
		1	0	1	1	=
	1	0	1	0	1	=
1	1	1	1	1	1	=

Converting from base-10 to base-2 (on the fly)

64	*32*	*16*	*8*	*4*	*2*	*1*		
							=	16
							=	55
							=	75
							=	84

Converting from base-10 to base-2 (using division)

- Repeatedly divide the decimal integer by 2. Each remainder is a binary digit in the translated value:

Division	Quotient	Remainder
37 / 2	18	1
18 / 2	9	0
9 / 2	4	1
4 / 2	2	0
2 / 2	1	0
1 / 2	0	1

37 = 100101

Addition

Base-10

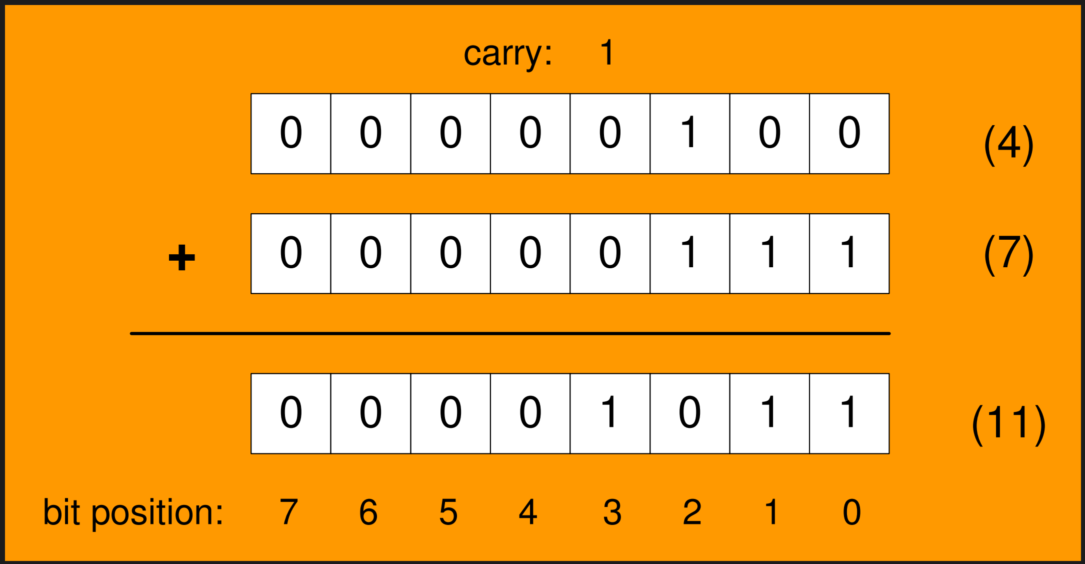
1	9	9	8
+		1	1
2	0	0	9

Base-2

1	0	1	1
+		1	1
1	1	1	0

Binary Addition

- Starting with the LSB, add each pair of digits, include the carry if present.



Practice binary arithmetic

1	0	1	1			1	1	1
+		1	1			+		1