Introduction to Computer Engineering ECSE-221

Introduction Class

Instructor

- Prof. T. Arbel
  - Email: WebCT
  - McConnell Eng Bldg room 425
- Office Hours:
    - Mondays, Wednesdays 12:30-1:30pm
    - McConnell Eng Bldg room 425
    - Also by appointment

Web Site

- WebCT:
  - http://www.mcgill.ca/webct/
  - Login should be set up the first time you use it.

The Course

- This course introduces the art of computer engineering.
- What a computing machine does (i.e. the operations that it performs) is defined by its architecture and low-level programming.
- These, in turn, are specified by Boolean operations, which are subsequently implemented by digital circuits.

The Course

- This course introduces the basic concepts and skills for:
  - digital circuit design,
  - low-level assembly programming,
  - computer architecture.

The Course

- This course introduces the background material on:
  - Boolean logic,
  - Data representation.

Pre-requisites

- COMP-202 – Introduction to Computing 1
  - We will assume some background knowledge on:
    - The design and implementation of programs using a modern high-level language, modular software design and debugging,
    - General programming concepts.

This Course Leads to…

- ECSE-322 Computer Engineering
- ECSE-323 Digital System Design

Which Leads To…

- ECSE-425 Comp. Org and Arch
- ECSE-427 Operating Systems
- ECSE-428 Software Engineering
- ECSE-525 Computer Architecture
- ECSE-526 Artificial Intelligence
- ECSE-531 Real Time Systems
- ECSE-532 Computer Graphics
- ECSE-543 Numerical Methods in EE
- ECSE-547 Finite Elements in EE

Learning Outcomes

- During this course, the student will acquire basic knowledge in, and should be able to apply, in a design context, the following aspects of computer systems:
  - Data representation in digital computers.
  - Boolean algebra.
  - Basic combinational circuits; their analysis and synthesis.
  - Elements of sequential circuits: latches, flip-flops, counters and memory circuits.
  - Computer structure, central processing unit, machine language.
  - Assemblers and assembler language.

Questions

- How are programs written in a high-level language, such as C or Java, translated into the language of hardware?
- How does the hardware execute the resulting program?
- Need to understand aspects of both hardware and software that affect program performance.

Questions

- What is the interface between software and hardware?
- How does software instruct hardware to perform needed functions?
- Key to understand how to write different kinds of software.

Questions

- What determines the performance of a program?
- How can programmer improve performance?
- Depends on:
  - Original program
  - Software translation of program into computer language
  - Effectiveness of hardware at executing program.

Questions

- What techniques can be used by hardware designers to improve performance?

Instructional Method

- Lectures (3 per week)
- 5 graded assignments (+1 not graded)
- Tutorials (2 identical 1 hour tutorials per week)

- WebCT Discussion Forum

Tutorial Times

(The same 1-hr tutorial will be held twice a week)

Option 1:

M	5:35pm-6:25pm, or
W	3:35pm-4:25pm

Option 2:

M	6:35pm-7:25pm, or
W	3:35pm-4:25pm

Option 3:

M	5:35pm-6:25pm, or
W	4:35pm-5:25pm

Option 4:

M	6:35pm-7:25pm, or
W	4:35pm-5:25pm

Tutor

- John Harrison: “John Harrison” on WebCT
- The tutor will be responsible for:
  - giving the class tutorials,
  - providing assistance for assignments and midterms,
  - overseeing the grading performed by the 6 Teaching Assistants.

Assignments

- The assignments will consist of written problems and laboratory work.
- Assignments will be handed out by posting on the course web page.
- Will have approximately 2 weeks.

Evaluation Method

- Assignments – 5 x 4% = **20%**
- MidTerm – **25%**
  - 1 hour in class time
  - Covers first 3 modules
- Final – **55%**
- *Tentative dates found on course outline!*

Textbooks and Course Materials

- A full set of notes for the course will be supplied.
- Textbook:
  - Patterson, David A., and Hennessy, John L., *Computer Organization & Design - The Hardware/Software Interface*, Third Edition, Morgan Kaufmann, San Francisco, 1998, ISBN 1-55860-428-6.

- Textbook is available at **McGill University Bookstore** and on reserve at the PSE library (Physical Sciences and Engineering Library).

Textbooks and Course Materials

- Not all of the text will be covered.
- Textbook provides a good overview of the field of Computer Engineering and pointers to many of the topics covered in later courses.
- Approximately 1/3 of the course is devoted to elementary digital system design. The Patterson and Hennessy text only covers this material briefly in Appendix B. The required material will be presented in class.

Textbooks and Course Materials

- In later courses (e.g. ECSE 323), students will have the opportunity to design, implement, and test digital systems in a laboratory.
- For ECSE 221, we will only be using a digital logic simulation tool to investigate the behaviour of these systems: LogicWorks.
- The program is installed on both the Department and Faculty computing facilities.

Textbooks and Course Materials

- For assembly programming, we will use a simulator for the MIPS R2000/R3000 machines described in the textbook: SPIM. It is installed in the Department and Faculty computing facilities.
- The program is also available for download at http://www.cs.wisc.edu/~larus/spim.html.

Textbooks and Course Materials

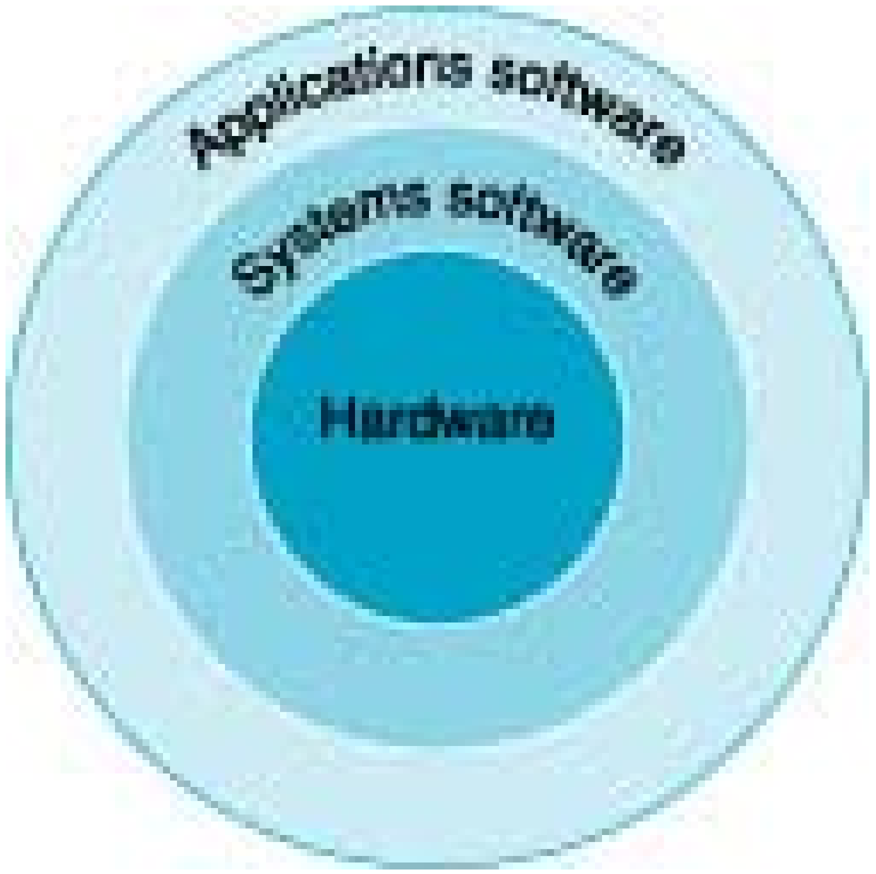
- For C programming, any “C” compiler under any operating system (DOS, Windows 9X/NT/2000, MacOS, UNIX) will do, provided that it is ANSI compliant (which includes most of the recent compilers).
- For those not familiar with “C” programming and debugging/development tools, tutorial sessions will cover the basics required for the course.
- Freeware system called lcc32 is available for download on the course web page.

Have Fun!

Computer System

- Typical application (e.g. word processor) –
  - 100,000 – 1,000,000 lines of code!
  - Rely on sophisticated software libraries that implement complex functions in support of application.
- Hardware in computer can only execute extremely simple, low-level instructions.
- Need several layers of software to translate high-level operations to simple computer instructions.

Software Layers



P& H, *Computer Organization and Design*

Software Layers

- 2 types of system software central to every computer:
  - Operating system
  - Compiler

Operating Systems

- Interface between user’s program and hardware
- Provides a variety of services
- Some important functions:

Operating Systems

- Interface between user’s program and hardware
- Provides a variety of services
- Some important functions:
  - Handling basic input and output operations

Operating Systems

- Interface between user’s program and hardware
- Provides a variety of services
- Some important functions:
  - Handling basic input and output operations
  - Allocating storage and memory

Operating Systems

- Interface between user's program and hardware
- Provides a variety of services
- Some important functions:
  - Handling basic input and output operations
  - Allocating storage and memory
  - Providing for sharing of many applications using it at the same time

Operating Systems

- Examples:
  - Windows
  - Linux
  - MacOS

Compilers

- Many operations
- One important function:
  - Translation of program written in a high language (e.g. C or Java) -> instructions that hardware can execute (assembly language statements)

Computer Language

- Language of computer – *binary numbers*
- binary digit or bit (0 or 1)
- Computers work based on *instructions* – collections of bits that computer understands:
  - e.g. 0001001010100010 might mean add 2 numbers!

Computer Language

- Need to have symbolic way to communicate these instructions
- *Assembler*: Program that translates symbolic version of instruction into binary.

Add B, C -> 0001001010100010

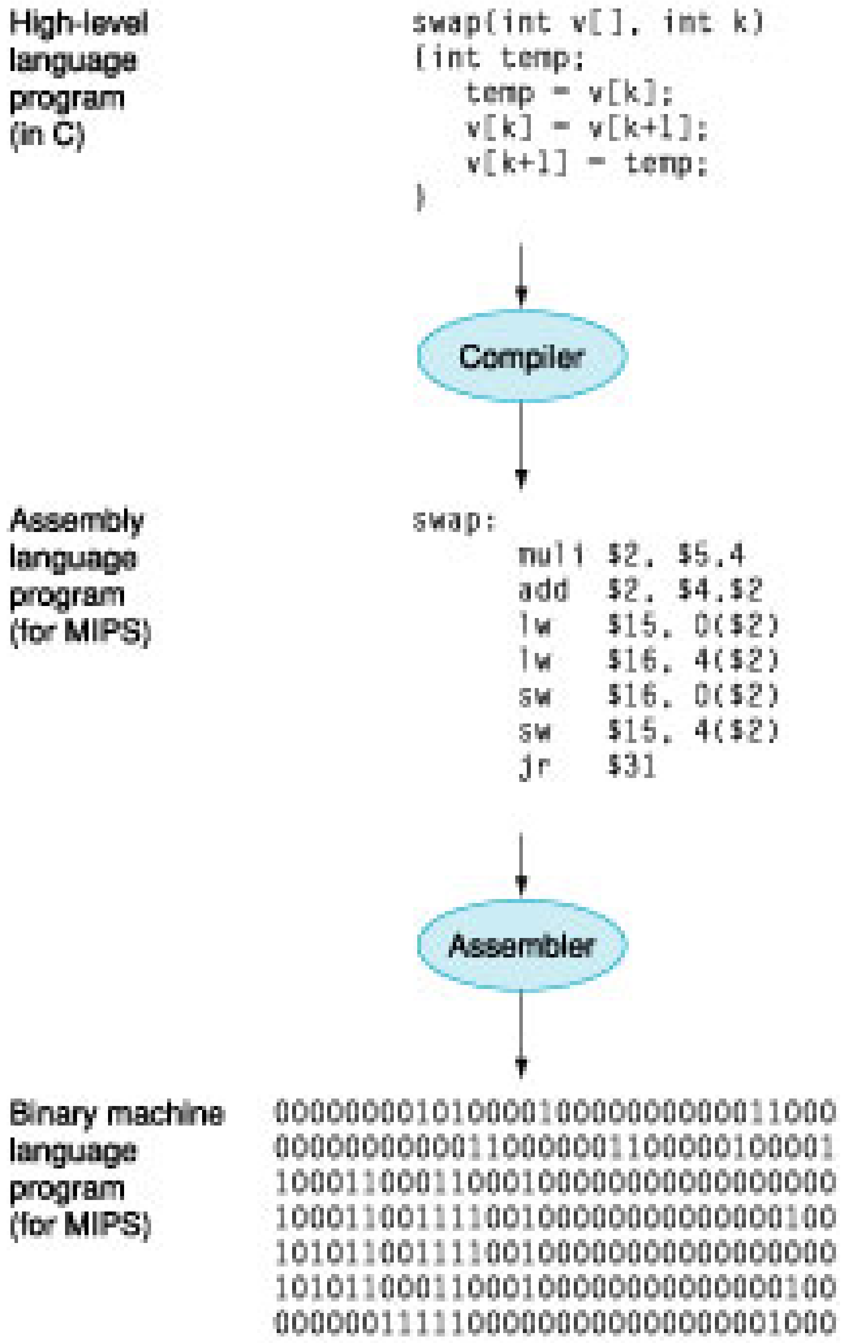
Computer Language

- Programming in assembly language means writing a line of code for each machine instruction!
- Needed more powerful language: *high-level programming language*.
- Compiler translates

B+C -> Add B,C

- Assembler translates

Add B, C -> 0001001010100010


High-level language program (in C)
Compiler
Assembly language program (for MIPS)
Assembler
Binary machine language program (for MIPS)

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

```
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
```

```
00000000101000010000000000011000
00000000000110000001100000100001
10001100011000100000000000000000
10001100111100100000000000000100
10101100111100100000000000000000
10101100011000100000000000000100
00000011111000000000000000001000
```

P& H, *Computer Organization and Design*

Computer Language

- Advantages of high level languages:
  - Allow programmers to think in more natural language
  - Allow languages to be designed according to use (e.g. Fortan: scientific computing)
  - Improve programmer productivity
  - *** Allow programs to be independent of compiler, hardware

Computer Engineering

- The underlying technologies are those of
  - electrical engineering
  - mechanical engineering
  - computer science
- What is a Computer System?
  - A structure for acquiring, storing, manipulating and delivering data

Computer Engineering

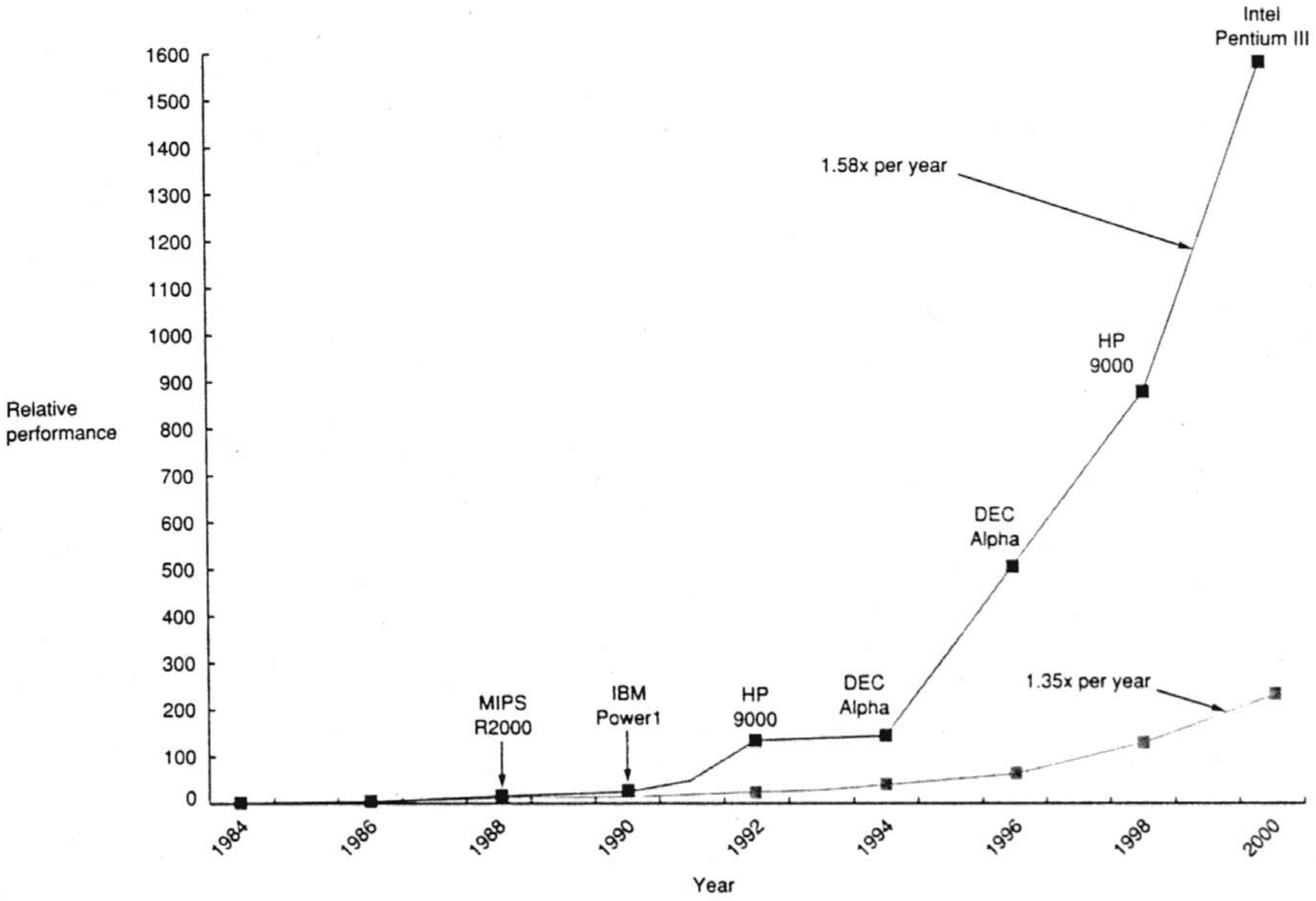
- The three major components are:
  - Communications
  - Storage
  - Processing

Computer Engineering

- The constraints on design are:
  - The current state of technology
  - Manufacturing capability
  - Cost
  - Reliability
  - Legal requirements
  - Physical size
  - ….

Computer Technology

- Computer technology has made incredible progress over 60 years.
- Overall progress is product of 3 factors:
  - Technology
  - Architecture
  - Compiler
- 1.6x/year improvement!
- No other technological system improves that fast (e.g. airplanes)


Intel Pentium III
1.58x per year
HP 9000
DEC Alpha
1.35x per year
MIPS R2000
IBM Power1
HP 9000
DEC Alpha
Relative performance
0
100
200
300
400
500
600
700
800
900
1000
1100
1200
1300
1400
1500
1600
1984
1986
1988
1990
1992
1994
1996
1998
2000
Year

Computer System

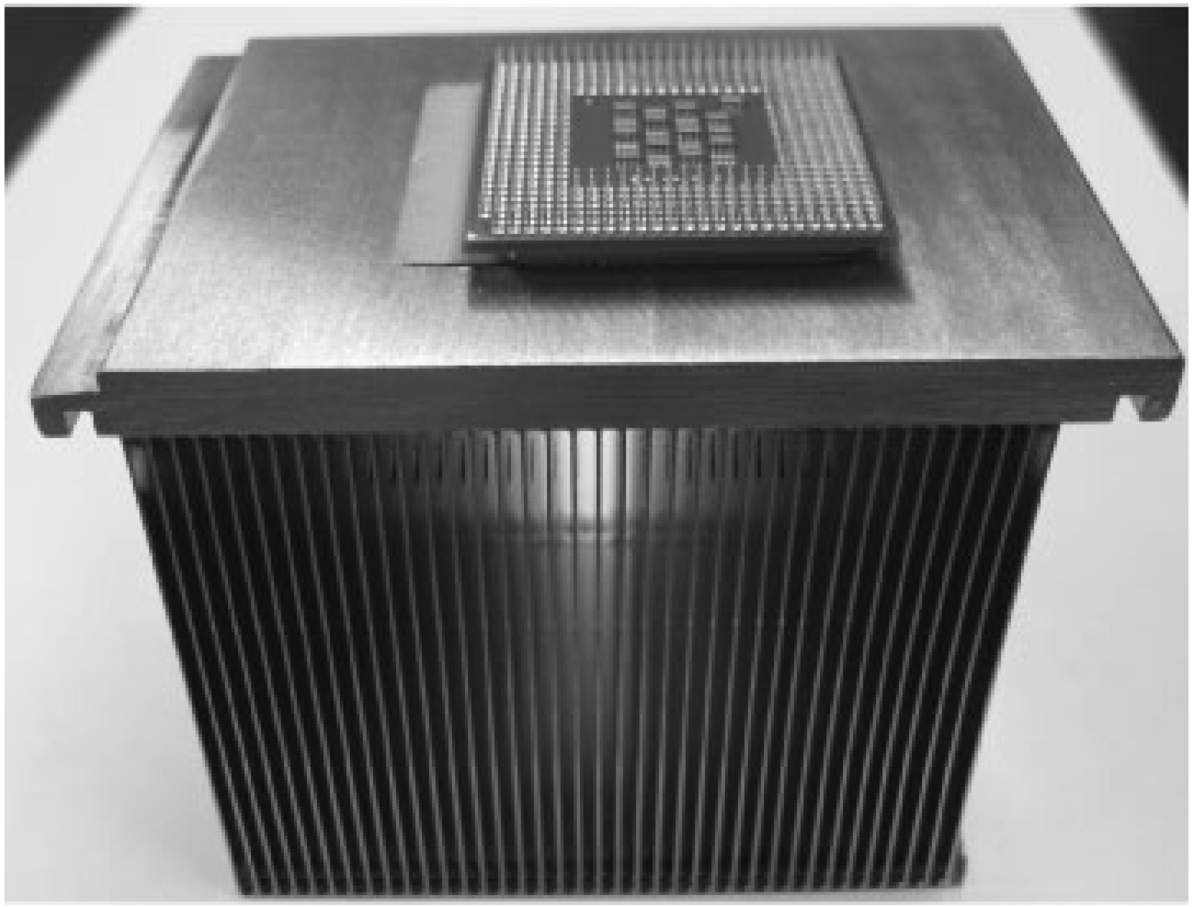
- Constructed from many devices
- CPU – *Central Processing Unit*
  - Digital logic for processing data via set of instructions
  - Early days: $$, huge. Today: cost decreased (commodity), why?
    - Advent of VLSI (Very Large Scale Integrated circuit)
    - Developments in IC production
  - Internal complexity, processing power increased dramatically:
    - room full of equipment -> chip a few centimeters square

Computer System

“Where… the ENIAC is equipped with 18,000 vacuum tubes and weighs 30 tons, computers in the future may have 1,000 vacuum tubes and perhaps weight just 1 ½ tons.”

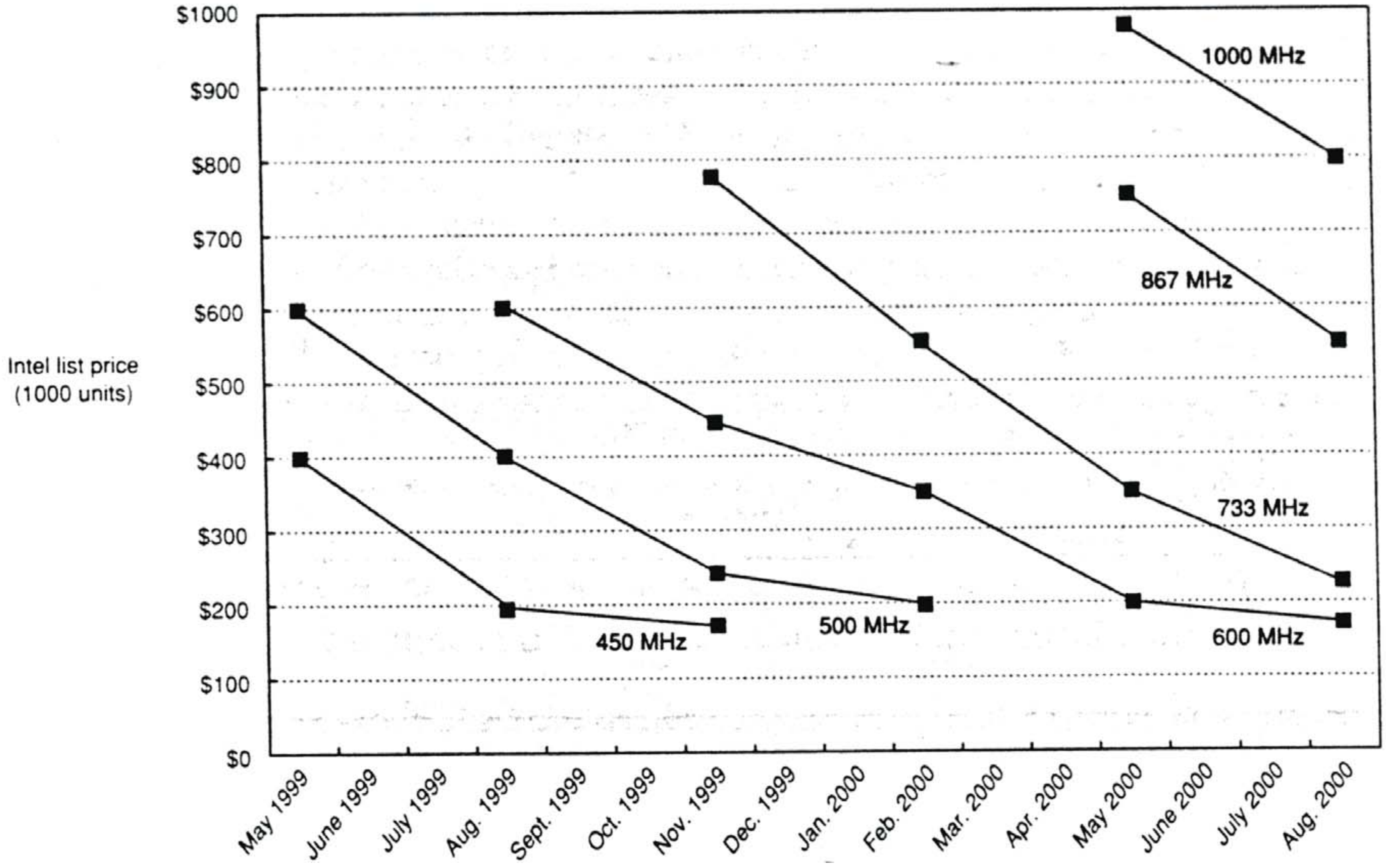
- Popular Mechanics, March 1949

Processors



Intel Pentium 4 (mounted on top of its heat sink)

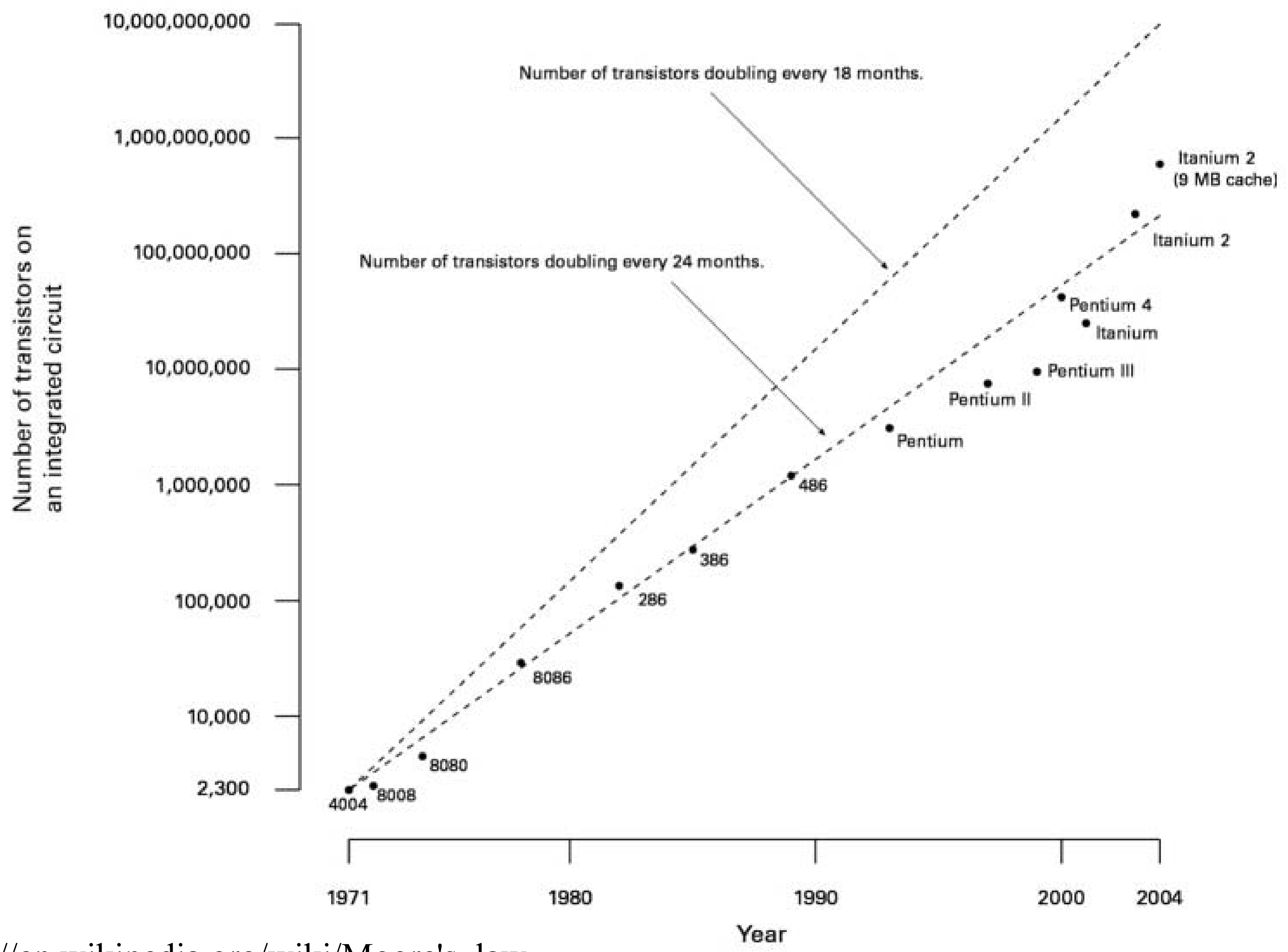
7 Generations of Pentium III chips



Moore’s Law

- **Moore's Law** is the observation that the complexity of integrated circuits, with respect to minimum component cost, doubles every 24 months (quoted as 18 months).
- It is attributed to Gordon E. Moore, a co-founder of Intel.
- If Moore's Law were applicable to the airline industry, a flight from New York to Paris in 1978 that cost $900 and took seven hours, would now cost about $0.01 and take less than one second.

Moore's Law

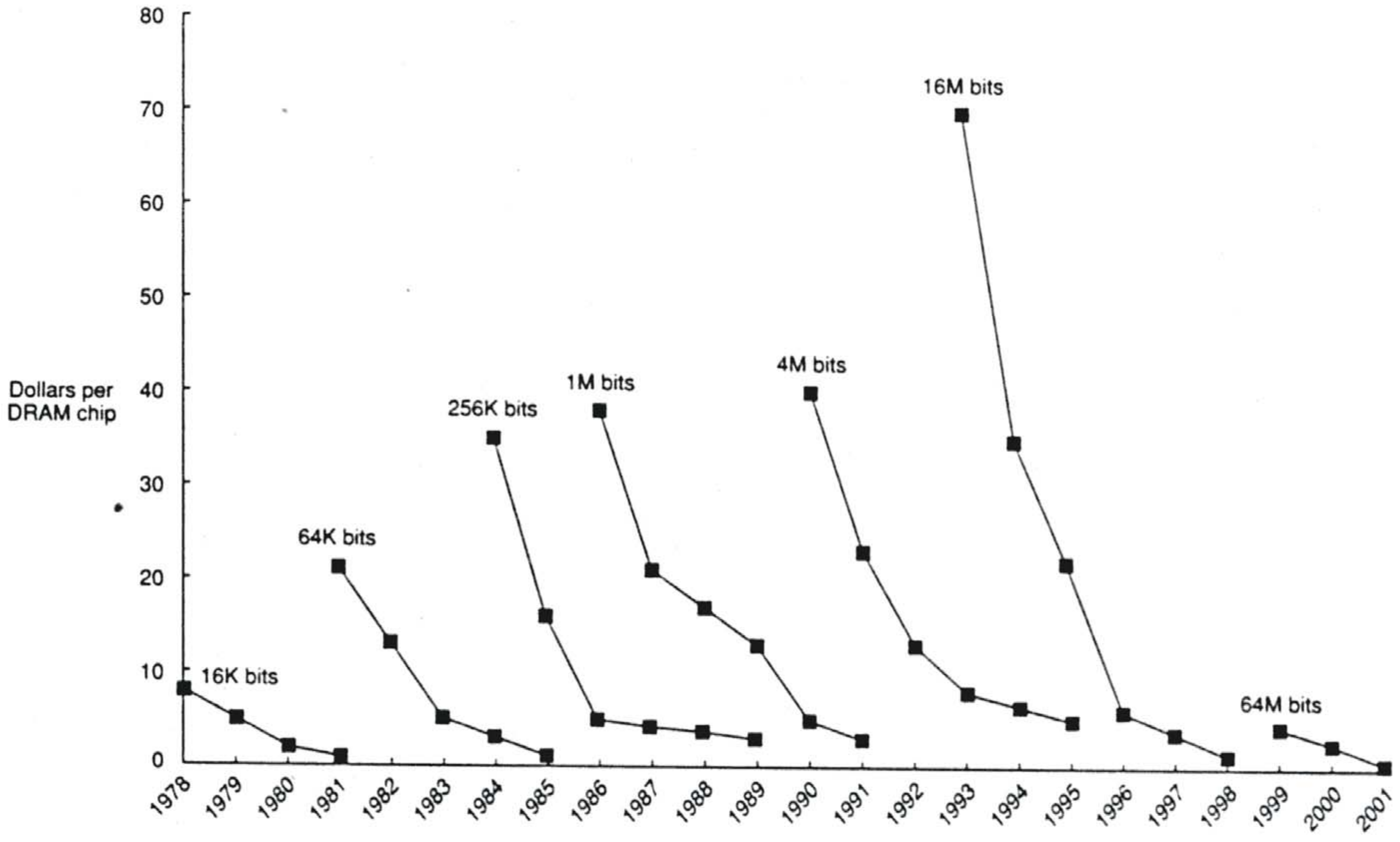


http://en.wikipedia.org/wiki/Moore's_law

System Components

- Developments in IC production - impacts on other parts of system:

- Memory costs have dropped dramatically over past 15 years (2-3 orders of magnitude)


Dollars per DRAM chip
80
70
60
50
40
30
20
10
0
16K bits
64K bits
256K bits
1M bits
4M bits
16M bits
64M bits
1978
1979
1980
1981
1982
1983
1984
1985
1986
1987
1988
1989
1990
1991
1992
1993
1994
1995
1996
1997
1998
1999
2000
2001

System Components

- Other components:
  - Input devices (e.g. keyboards, mice)
  - Output devices (e.g. monitors, printers)
  - Storage systems (e.g. memory, disks, tape)
  - Communications (e.g. modems, networking)

ECSE-322

Aim of 221 Course

- Provide sufficient background to understand fundamental operations of basic computer building blocks
- Introduce students to tradeoffs in computer design

Aim of 221 Course

- Introduce data representation, Boolean algebra
- Low level language -> defined by Boolean operations -> implemented by digital circuits
- Describe some basic architecture and assembly

Aim of 221 Course

- Provide strong basis which will be assumed and used extensively in subsequent courses!
- Material covered in this course required knowledge for computer design: used in today’s computers (e.g. Intel, graphics processors)