LIS 386.13
Information Technologies and the Information Professions

Introduction to Computing

R. E. Wyllys

Last revised 2002 Dec 9

Lesson Objectives

- You will
  - Understand how the basics of how computer hardware and software work
  - Understand the essential elements of a computer program
  - Become familiar with highlights of the history of computing

What Are Computers?

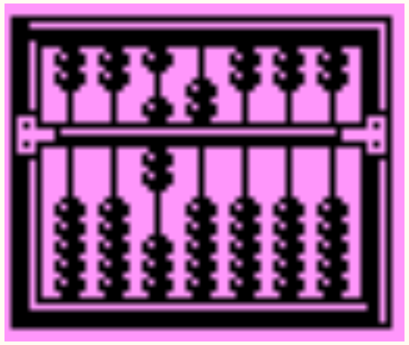
- Computers are a combination of
  - Hardware, i.e., physical units that use electrical, magnetic, and optical means to carry out various operations
  - Software, i.e., instructions that control the operations performed by the hardware
- The combinations are highly integrated.
- The operations are carried out at enormously high speeds (in comparison with the speeds of human activities).

The Secret to Computers

- Half of the secret to computers is the speed with which they carry out their operations.
- The other half of the secret is the fantastic flexibility and variety of the operations that computers can be directed to perform through the instructions of clever humans.

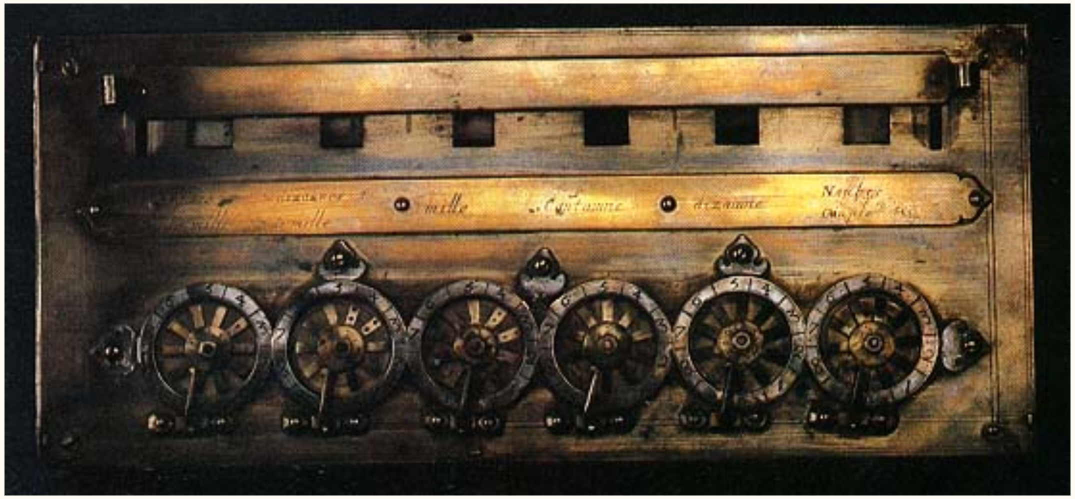
An Outline History of Computing

- The first invention that can be justifiably placed under the rubric of "computer" is the abacus
- The abacus was invented in China c. 2600 BC. It spread to many other cultures, and is still used today to assist arithmetic calculations.
- The upper sliding rings represent 5, the lower 1, and the rings have place values starting from the right side (just like Hindu numerals).



An Outline History of Computing

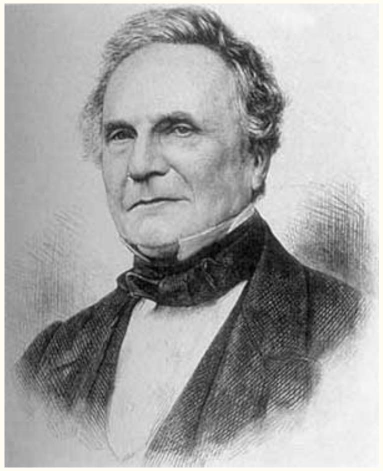
- In 1644 Blaise Pascal (1623-1662) invented a mechanical adding machine, using dials connected to toothed gears that operate like modern odometers.



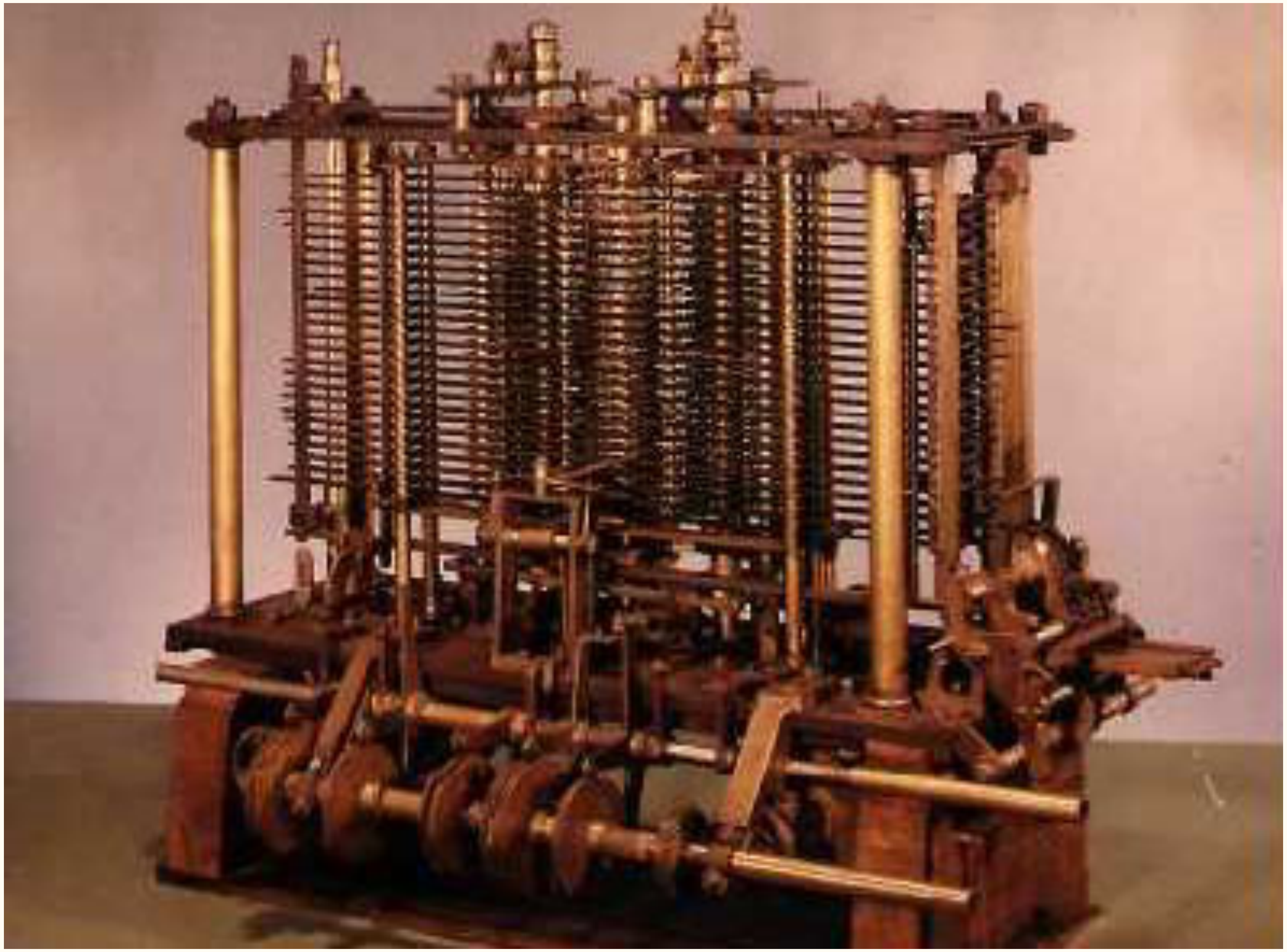
- In 1670 Gottfried Leibnitz (1646-1716) modified Pascal's device so that it could also multiply and divide.

An Outline History of Computing

- A closer approach to modern computers was the invention by Charles Babbage (1791-1871) in the late 1830s of the Analytical Engine, which carried out numerical calculations that could be programmed like those of a computer.
  - Babbage never succeeded in completing the full construction of his Analytical Engine, but his design was sound, and working models of it have been built in recent decades.
  - Ada Gordon King (1815-1852), Countess of Lovelace (and daughter of the poet Lord Byron), wrote sets of instructions (i.e., programs) for the Analytical Engine, and can justly be called the first computer programmer.



Babbage's "Analytical Engine"



An Outline History of Computing

- What can be viewed as the next step toward today's computers was the invention by Herman Hollerith (1860-1929) of electrical counting procedures and machines, using cards in which holes were punched, in columns, representing numbers.
  - Hollerith invented these for use in the 1890 census.
  - In 1896 he founded a company to manufacture his machines. The company was the direct ancestor of today's IBM Corporation.

An Outline History of Computing

- In the 1920s and 1930s Electrical Accounting Machines (EAMs) were developed by IBM (and others) that greatly speeded up punched card processing.
- In concurrent developments, machines using large numbers of telephone relays (10-position switches that made dialing a phone number possible) were developed to carry out ordnance-table computations (used in controlling the firing of cannons) for the U.S. Army.

An IBM Model 82 Punched Card Sorting Machine. It could process 650 cards per minute.

An Outline History of Computing

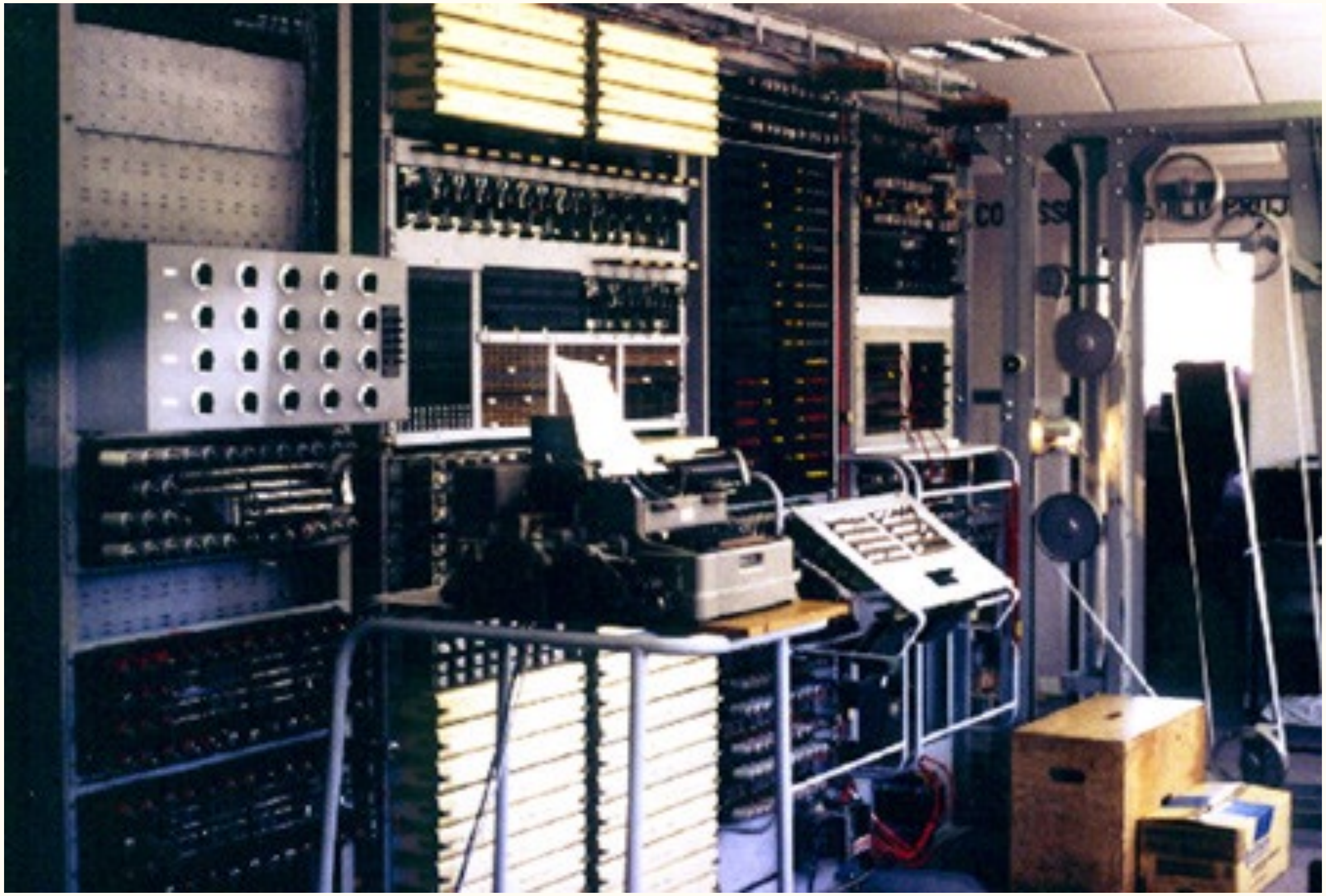
- EAMS were made to carry out varying sets of operations via plugboards.
- A plugboard's back consisted of a matrix of pins (connected to socket holes on the board's front), which fitted into a receptacle on the EAM.
- EAM operators would set up a plugboard for a particular operation by connecting appropriate pairs of socket holes with jumper wires. Once set up for a given operation, a plugboard could be stored between uses, without having to be re-plugged. Changing an EAM from one operation to another could be done quickly and easily, because exchanging plugboards in the EAM receptacle took only a few seconds.

An Outline History of Computing

- During World War II, British cryptologists at their headquarters in Bletchley Park invented, in 1943, the first true electronic computer, Colossus, to aid in the decryption of German radio messages.[1]
- The development of Colossus was based partly on the success of EAMs and ordnance-calculation machinery, but also partly on the work of a mathematical genius, Alan Turing (1912-1954), who was on the Bletchley Park staff.
  - Using 1500 vacuum tubes, Colossus carried out instructions, from programs applied through switches set by operators, on data input via punched paper tape. The tapes were read at rates as high as 5000 characters/second, to reach which rates the paper tape traveled at 30 mph (49km/h).

[1] Colossus, Bletchley Park, and much else are explained well in: Budiansky, Stephen. Battle of Wits: The Complete Story of Codebreaking in World War II. New York, NY: Free Press; 2000. ISBN:0-684-85932-7.

An Outline History of Computing



The reconstruction of Colossus at the Bletchley Park Trust Museum in the U.K.

An Outline History of Computing

- The British cryptologists shared their invention with their U.S., Canadian, and Australian allies. Consequently, when the end of WWII brought an end to the ultra-tight security surrounding Colossus, the fact that electronic computers worked became widely known.
  - Even though Colossus had worked with letters, most initial ideas for using computers centered about their ability to do arithmetic extremely fast.
  - But some people with vision, such as Vannevar Bush (1890-1974) and John von Neumann (1903-1957), suggested in the late 1940s that computers might be used to do such tasks as machine translation of languages (an idea that actually took over 35 years to reach fruition).

An Outline History of Computing

- ENIAC (Electronic Numerical Integrator and Computer) was the first programmable computer to gain wide public recognition.
  - ENIAC was built at the University of Pennsylvania under the direction of J. Presper Eckert (1919-1995) and John W. Mauchly (1907-1980).
  - The ENIAC project was made public in 1946, although it began in 1942.
    - The first version (a pilot version) of ENIAC became operational in July 1944.* This was 6 months after Colossus was already in full operation, making Colossus the first true digital computer in the world.

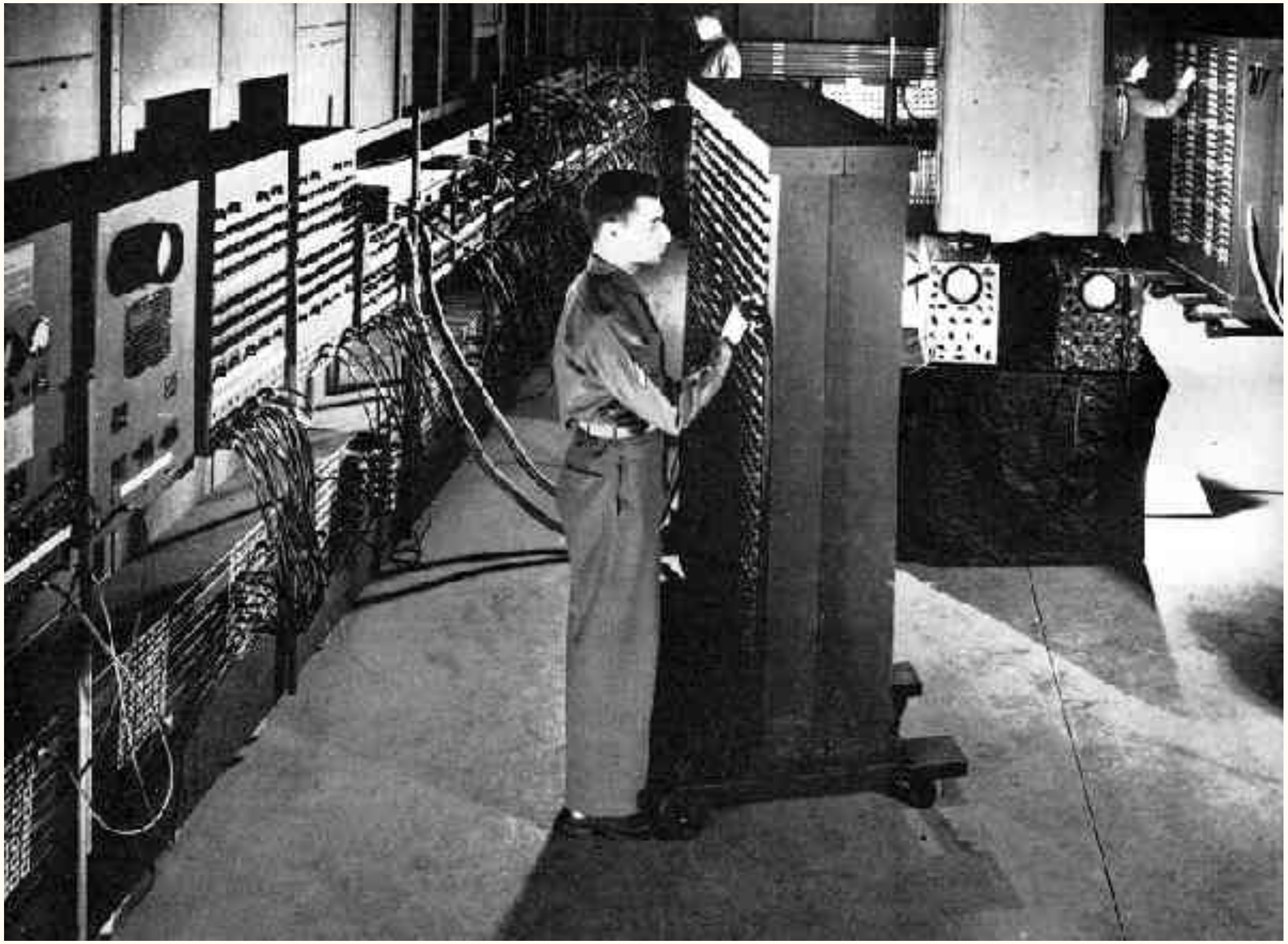
*Williams, M. R. History of Computing Technology. 2nd ed. Los Alamitos, CA: IEEE Computer Press; 1997. P. 281.

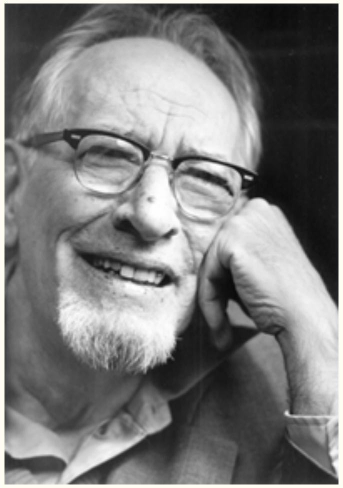
An Outline History of Computing

- ENIAC used 17,000 vacuum tubes, 70,000 resistors, and 6000 switches. It operated at a rate of 5000 additions and/or subtractions per second.
  - Its clock speed was probably equivalent to about 20,000 cycles per second, i.e., 20KHz.
- From today's viewpoint, an interesting aspect of ENIAC's development was that though six women* were the original programmers of ENIAC, the news media of the time ignored that fact. As a result, the public gained the impression that ENIAC's operation, and computing in general, was an entirely masculine endeavor.
- The following slide shows a picture of ENIAC (a woman is in the background), together with pictures of J. Presper Eckert (upper right) and John W. Mauchly (lower right).

*Light, Jennifer S. When Computers Were Women. Technology and Culture. 1999 July, 40(3):455-483.

An Outline History of Computing





An Outline History of Computing

- An example of how hard it is to predict the future of technology is the following quotation* from the March 1949 issue of *Popular Mechanics* magazine:
  - "Where a calculator like the Eniac today is equipped with 30,000 vacuum tubes and weighs 30 tons, computers in the future may have only 1,000 vacuum tubes and perhaps weigh only 1 ton."

*This quotation appeared as the "Resurrected Quote of the Week" on the editorial page of the *Richmond Times-Dispatch*, April 21, 2001.

An Outline History of Computing

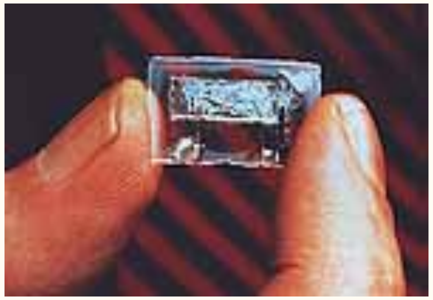
- Though commercially built computers were bought by the Bureau of the Census in time to be used in the 1950 census, the first really successful business uses of computers began with the development of the IBM 701 in 1951.
- By the early 1960s businesses, especially financial institutions, were already heavily dependent on electronic computers.
- That dependence has intensified ever since; it now pervades all of the developed world; and it has extended its scope from large businesses and government agencies to individual people from ages 5-6 to 100+.

Development of Computer Hardware

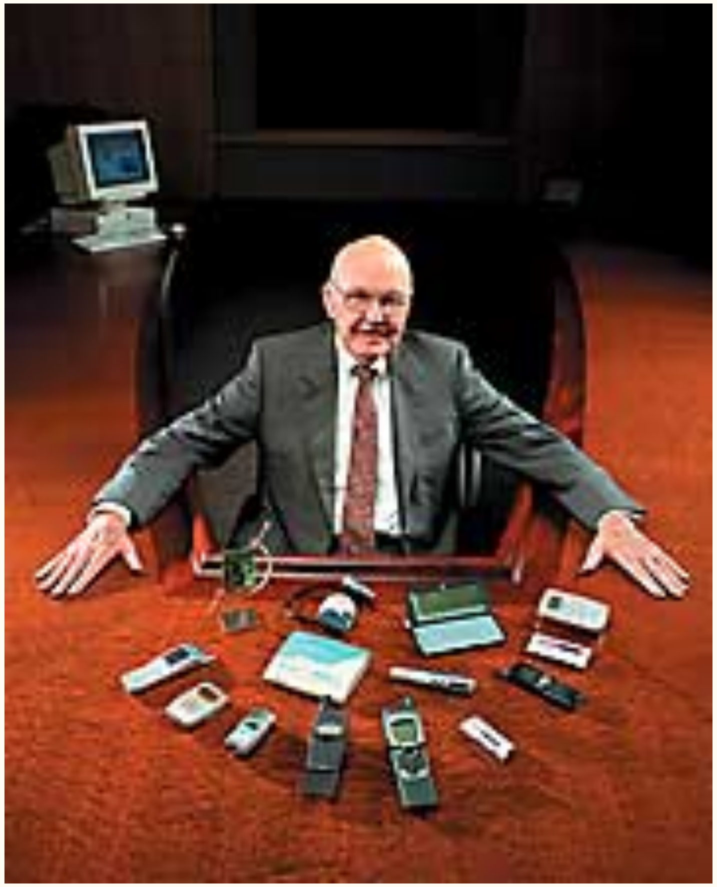
- Babbage's Analytical Engine (1830s) used mechanical gears.
- Electric accounting machines and ordnance calculators (1920s - early 1940s) used electrically operated mechanical relays.
- Colossus (1943) used vacuum tubes, and thus demonstrated the vastly greater speeds of non-mechanical electronic operations
- Vacuum-tube circuitry was used in all computers of the 1940s and most of the 1950s.
- By 1958 transistors, invented in 1948, had been improved to the point where the first transistorized computers reached the market.

Development of Computer Hardware

- In 1958-1959, Jack Kilby (1923-), working at Texas Instruments in Dallas, and Robert Noyce (1927-1990), working at Fairchild Semiconductor in San José, independently invented the monolithic integrated circuit: a way of miniaturizing transistors and related elements of electronic circuits and packing large numbers of them very close together.



Kilby’s first integrated circuit



Jack Kilby and devices that use integrated circuits

Development of Computer Hardware

- For the invention of the integrated circuit, Kilby was awarded the Nobel Prize in Physics in 2000. Sadly, Noyce had died in 1990, while serving as President of Sematech in Austin, and thus missed becoming a Nobel Laureate.
  - Kilby has often been called "The Texas Edison." He was a co-inventor of the hand-held electronic calculator and of the first thermal printer, holds over 60 U.S. patents, and has been awarded numerous other honors in addition to the Nobel Prize.
- Kilby's Nobel Lecture provides a fascinating overview not only of the invention of the monolithic integrated circuit but also of the extraordinary changes and developments in electronics over the last 40 years.

Development of Computer Hardware

- The monolithic integrated circuit (IC), together with its predecessor invention, the transistor, are unquestionably the two most important inventions of the 20th century, having brought about enormous changes in the way the world works. The IC has made possible the amazing reductions in the size of electronic circuits that have led to desktop computers, cellular telephones, personal digital assistants, communications satellites, in-ear hearing aids, global positioning systems, etc.
- Typical transistors of the 1950s were cylinders about 0.5cm to 1cm in length and 0.5cm in diameter. Current CPU chips contain the equivalent of about 20 million transistor circuits in a chip about 10cm long, 4cm high, and 0.5cm thick.
- Dense packing of circuits not only reduces overall size but also means electrons travel shorter paths, thus increasing the speed of operations.

Development of Computer Hardware

- A personal experience of hardware miniaturization
  - I was working for System Development Corporation in 1963 when SDC acquired one of the nine units ever manufactured of what was the most powerful computer in existence at the time, the IBM STRETCH. It filled a room about 40' x 40' (12m x 12m).
  - In 1983, just 20 years later, I acquired my first desktop computer. With the same physical dimensions as current PCs, it was more powerful than SDC's room-sized, world's most powerful computer of 1963.
  - My 1983 PC had 256KB of RAM (Rapid Access Memory) and a 10MB hard drive, and ran at a clock speed of 4.7MHz. It had no mouse, and its screen displayed only alphanumeric characters, i.e., no graphics, showing amber letters on a black background.
  - My current PC has 512MB of RAM and 42GB of hard drive space, and runs at a clock speed of 850MHz. In comparison with the same features in my PC of 19 years ago, the ratios are 2000:1, 4200:1, and 181:1, respectively.

Development of Computer Hardware

- Another comparison sheds light on costs. In 1975 F. P. Brooks* wrote:
  - "Aside from running time, the space occupied by a program is a principal cost. . . .Consider the IBM APL interactive software system. It rents for $400 per month and, when used, takes at least 160 K bytes of memory. On [an IBM Model 360/165], memory rents for about $12 per kilobyte per month. If the program is available full-time, one pays $400 software rent and $1920 memory rent for using the program. . . .

*Brooks, Frederick P., Jr. (1995). *The Mythical Man-Month: Essays on Software Engineering: 20th Anniversary Edition*. Reading, MA: Addison-Wesley. ISBN: 0-201-83595-9. [This edition is a re-issue, with additional material, of the initial publication in 1975.]

Development of Computer Hardware

- "One frequently hears horror expressed that a 2 M byte machine may have 400 K devoted to its operating system. This is as foolish as criticizing a Boeing 747 because it costs $27 million. One must also ask, 'What does it do?' What does one get in ease-of-use and in performance (via efficient system utilization) for the dollars so spent? Could the $4800 per month thus invested in memory rental have been more fruitfully spent for other hardware, for programmers, for application programs?"

- You should note that Brooks states costs in 1975 dollars. His figures need to be multiplied by about 3.3 to provide figures in 2001 dollars. Do that (e.g., 3.3x$4,800 = $15,840 monthly memory rental), and then think about the size of your microcomputer's memory and the cost today of purchasing (not just renting) RAM. That comparison will give you the beginnings of an understanding of how computer costs have decreased over the last 20-odd years.

Development of Computer Hardware

- Still another comparison of hardware sizes and speeds
  - In November 2000, IBM announced its receipt of a contract from the U.S. Navy to build a supercomputer that would perform 480 billion calculations per second, using 320 microprocessors (size unspecified), 224GB of RAM, and 2.9TB of disk storage.
    - Using the currently reasonable estimate that each of the 320 microprocessors contains the equivalent of 20 million transistors, we can assess this supercomputer's CPU as having the equivalent of 6.4 billion transistors, or approximately 376,000 times the CPU resources of the ENIAC.
    - In terms of calculations per second, this supercomputer is about 480,000,000,000/5,000 = 96,000,000 times more powerful than the ENIAC.

Development of Computer Hardware

- And the supercomputer race continues.
  - In May 2001 IBM announced a contract for the delivery, by the end of 2002, a supercomputer capable of 3.8 trillion calculations per second.
    - For details, see http://www.zdnet.com/eweek/stories/general/0,11011,2761604,00.html
  - A Website that keeps track of the 500 fastest supercomputers in the world (at any given time) is

    http://www.top500.org/

Development of Computer Hardware

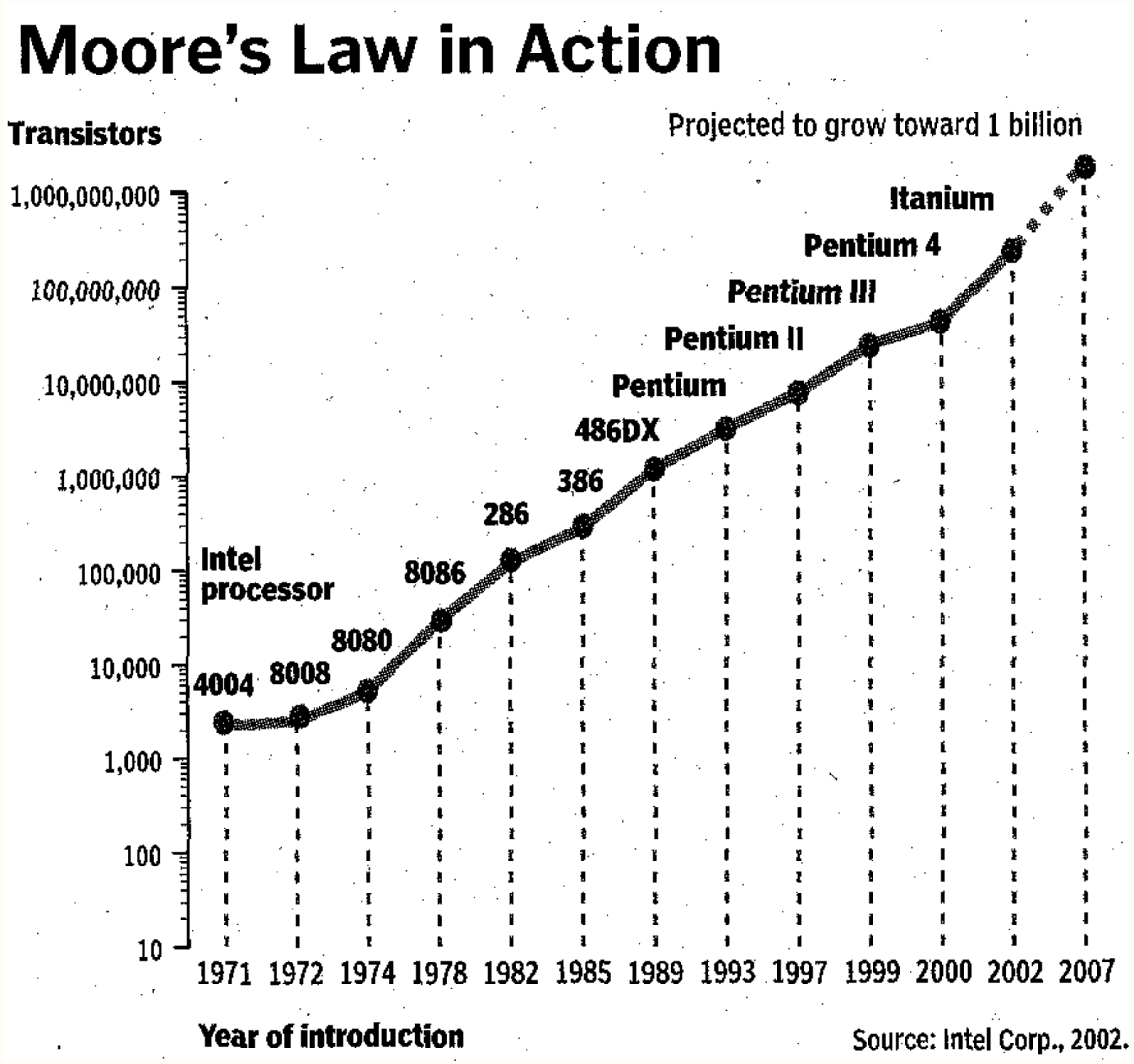
- In 1965 Gordon Moore (1929-) (who, along with Robert Noyce and Andrew Grove [1936-], would found Intel in 1968) made an observation that has become famous.
- He had noted that the computing power that could be built into a single IC had doubled about every 18 months for several years before 1965. He predicted that that rate of increase would continue.
- This prediction, ***Moore's Law***, has held for over 30 years now, and is expected to hold for at least another decade.
  - An interesting recent review is "The Lives and Death of Moore's Law" by Ilkka Tuomi.

The picture shows, left to right, Andrew Grove, Robert Noyce, and Gordon Moore.

Moore's Law

- The slide that follows shows one way in which Moore's Law operates: viz., in terms of the number of transistors that can be placed on one integrated circuit chip.
- The graph shows the growth of this number
  - – 2,300 transistors on an Intel 4004 chip in 1971
  - – 29,000 on an Intel 8086 chip in 1978
  - – 3.1 million on the first Pentium chip in 1993
  - – 7.5 million on the Pentium II chip in 1997
  - – 42 million on the Pentium 4 chip in 2000
  - – 300 million on the Itanium chip in 2002
  - – 1 billion projected on the Intel chip of 2007

Moore's Law



From: PC Magazine, April 23, 2002, p. 6

Computers Are a Combination of Hardware and Software

- Computers function by combining the capabilities of hardware and software
  - Basic types of hardware
    - Input devices
    - Output devices
    - Storage devices
    - Central Processing Units (CPUs)
  - Basic types of software
    - Operating system programs
    - Applications programs
    - Utility programs

Hardware

- Input devices include
  - Keyboard
  - Mouse
  - Scanner
  - Bar-code reader
  - Floppy disk, CD, DVD, and tape drives
  - Modem
  - Network interface
  - Microphone
  - Joystick

- Output devices include
  - Monitor
  - Printer
  - Floppy disk, CD-R, and tape drives
  - Modem
  - Network interface
  - Speakers

Hardware

- Storage devices include
  - Cache memory (ultra high-speed)
  - Random-access memory (RAM) (high-speed)
  - External units (low-speed), such as
    - Floppy disks
    - Zip-drive cartridges
    - CDs and DVDs
    - Tapes
    - Networked storage
    - World-Wide Web (WWW) storage

Hardware

- Central Processing Unit
  - Uses operating system (OS) to
    - Handle the priorities of operations and the interleaving of multiple tasks and/or users
    - Work its way through a sequence of instructions in each user's program
    - Move data back forth among CPU, input and output (I/O) units, RAM, hard-disk storage, and external storage
  - Carries out arithmetic operations and logical tests in order to perform the operations required by users' programs

Software

- Operating systems (OSs), via central processing unit (CPU),
  - Handle the priorities of operations and the interleaving of multiple tasks and/or users
  - Work their way through sequences of instructions in the programs that are running
  - Move data around as needed within the computer and its peripheral devices
- Examples of OSs
  - Microsoft Windows 98, Millennium, NT, 2000, CE
  - Macintosh Operating System
  - Unix and Linux
  - BeOS

Software

- Operating systems (OSs)
  - Can be classified according to their ability to perform few or many functions essentially simultaneously
    - Single-user, single-task OSs: Apple DOS, CP/M, Microsoft DOS
    - Single-user, multiple-task OSs: Windows 98, Millennium, CE; Macintosh OS
    - Multiple-user, multiple-task OSs: Unix, Linux; Windows NT and 2000 (to limited extents)

Software

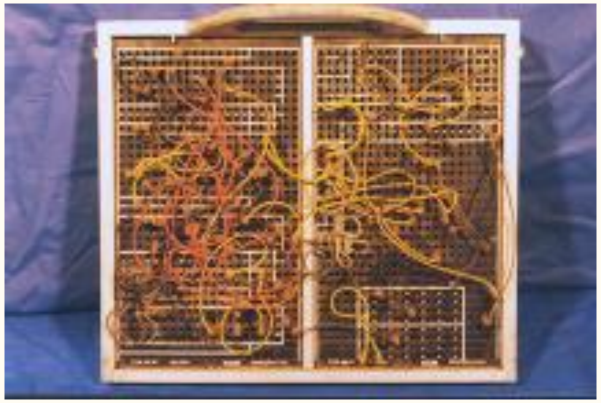
- Application programs (Apps)
  - Perform related sets of (major) tasks that users desire to perform
    - The variety of things application programs do is limited only by programmers' imaginations and by users' needs and willingness to buy the programs
  - Examples
    - MS Word, Excel, Access, Outlook, PowerPoint
    - Web Browsers: Netscape, Internet Explorer
    - Adobe Photoshop (image manipulation)
    - Dream Weaver (Webpage development)
    - Quicken Financial Suite, TurboTax
    - OmniPage (optical-character recognition, OCR)
    - Games and Simulation Programs: Everquest, Flight Simulator, Quake, SimCity, Solitaire

Software

- Utility programs
  - Perform specialized sets of (minor) tasks that aid users in using the applications programs
  - Examples
    - Character Map (provides easy access to special characters and non-English alphabets)
    - Dial-Up Networking (handles modems and telephone-line connections)
    - NetMeeting (handles interactive messaging)
    - PrintScreen (captures to clipboard, and prints, contents images of full monitor display or windows within the display)
    - Telnet (handles uploading and downloading of files between desktops and Internet sites)
    - Windows Calculator (handles simple arithmetic calculations)
    - WinZip (compresses and decompresses files)

History of Programming Languages

- Ordnance-calculation machines, Colossus, and other early computers were controlled (i.e., programmed) by manual setting of toggle switches and by hand-wiring of plugboards.
  - Plugboards were electrical boards with copper pins on one side and, on the other side, holes into which hundreds of connector cables could be plugged as needed for various starting conditions.
  - Some of the toggle switches and plugboards were used in ways analogous to today's OSs; others, in ways analogous to today's application programs.



History of Programming Languages

- Control sequences (programs) for computers in the latter half of the 1940s and the early 1950s were punched into punched cards and paper tape in the form of
  - Sequences of 0s and 1s at first, but soon, because binary numbers are so difficult for humans to handle, in the form of octal numbers each of which represented three binary digits.
  - Binary (and octal, and nowadays, hexadecimal) coding is called ***machine language***, because it is directly readable by CPUs

History of Programming Languages

- As a simplified but suggestive example of machine-language (binary-level) programming, suppose that in an instruction in a computer program
  - The first three binary digits represent the machine operation to be performed
  - The second three binary digits represent the address of the memory location used in the operation
- Suppose also that a particular working area of the CPU is called the "Accumulator"

History of Programming Languages

- Then, in our example of machine-language programming, we might write
  - 001101 to represent "Load Accumulator with contents of memory cell 5" (Note*: $101_2 = 5_{10}$)
  - 010111 to represent "Add to Accumulator the contents of memory cell 7" (Note: $111_2 = 7_{10}$)
  - 011101 to represent "Store Accumulator's contents in memory cell 5"
- The result of these operations would be to use the Accumulator to add the starting contents of memory cells 5 and 7, and store the resulting sum into memory cell 5.

*The subscripts denote the type of arithmetic: 101_2 means "101 in binary (i.e., base 2) notation"; 5_{10} means "5 in decimal (i.e., base 10) notation".

History of Programming Languages

- Around 1950 people realized that it would be possible to represent the binary digits that determined the actual machine operations by mnemonic codes that would be much easier for humans to handle.
- The mnemonic-code level of programming language is called ***assembly language***.
- Assembly languages required that special programs, called ***assemblers***, be used to translate the assembly-language instructions into binary sequences.
- Assembly-language programming is still occasionally used in special circumstances.

History of Programming Languages

- As an example of assembly-language (mnemonic-code) programming, we might write
  - LDA05 to stand for "Load Accumulator with contents of memory cell 5"
  - ADA07 to stand for "Add to Accumulator's contents the content of memory cell 7"
  - MLA06 to stand for "Multiply Accumulator's contents by contents of memory cell 6"
  - STA05 to stand for "Store Accumulator's contents into memory cell 5"
- The result of these operations would be to add the starting contents of memory cells 5 and 7, then multiply that sum by the starting contents of memory cell 6, and finally, store the resulting product into memory cell 5.

History of Programming Languages

- From the human viewpoint, assembly-language programming was a great improvement over binary-level programming, but it was still very tedious.
- The next step in making programming easier for humans was the development, starting in the mid-1950s, of what are called ***procedure-oriented*** or ***procedural languages***.

History of Programming Languages

- As an example of a procedural language, we might write statements like
  - X = 5
  - Y = 7
  - Z = X + Y
- In this example, the variable X is set equal to 5; the variable Y is set equal to 7; and then variable Z is defined as the sum of variables X and Y.
- The above example looks much like the way such instructions would be written in Algol or FORTRAN. In COBOL, comparable instructions would look like
  - X EQUALS 5
  - Y EQUALS 7
  - Z EQUALS 5 PLUS 7

History of Programming Languages

- Among the first procedural languages were Algol (Algorithmic Language), FORTRAN (Formula-Translation Language), and COBOL (Common Business-Oriented Language).
- Procedural languages require that special programs, called ***interpreters*** and ***compilers***, be used to translate the procedural statements into binary sequences.

History of Programming Languages

- Credit for writing the first compiler is generally given to Grace Hopper (1906-1992), as part of the development of COBOL, in which she played a leading role.
  - Additional pictures of Rear Admiral Hopper are available at http://www.history.navy.mil/photos/pers-us/uspers-h/g-hoppr.htm

History of Programming Languages

- Procedure-oriented languages are clearly much more convenient for humans than assembly languages (let alone binary-level languages).
- Developments since the 1950s have emphasized making procedural-type languages ever more powerful.

History of Programming Languages

- Current procedural languages include C++, Java, and Visual Basic.
  - As an example, here are four lines from a program written in Visual Basic for Applications for Excel:
    - Range("A1:A25").Select
    - Selection.Copy
    - Windows("rethldgc.xls").Activate
    - Range("U5").Select
  - These lines select a range of cells in an Excel worksheet, copy their contents, activate another worksheet, and select a cell in the new worksheet.

History of Programming Languages

- In recent years, programming languages have developed in the direction of object-oriented programming.
  - In the programming world, what is meant by the term ***object*** is a set of data together with some programmed code that can manipulate the data.
  - An example might be a table of data in a Microsoft Excel spreadsheet that is copied into a MS Word document, in such a way that when the data are updated in Excel, the corresponding table in the Word document is also, and automatically, updated.

History of Programming Languages

- The general idea is that once an object has been defined in a particular object-oriented program (OOP), the object can:
  - Be used repeatedly and easily within the program
  - Have sub-objects, which inherit (i.e., share) all the properties of the parent object but also have specific properties of their own
- Object orientation has proven a very powerful approach to programming and has greatly speeded up the writing of large, complex programs.
- C++ and Java are examples of OOPs.

Markup Languages

- Related to programming languages are ***markup languages*** (MLs). MLs
  - Were originally developed as a tool by which editors of books and articles could furnish instructions to typesetters about text formatting, arrangement of tables and equations, placement of illustrations, etc.
  - Have grown to include instructions for defining types of data (e.g., author name, editor name), elements of an outline (e.g., title, section heading, sub-heading), etc.

Markup Languages

- The markup-language idea has been unified by the Standard Generalized Markup Language (SGML), an international standard established by the International Organization for Standardization (ISO).
  - SGML uses codes enclosed in angle brackets. Most codes are used in pairs: The combination <*code*> marks the beginning, and </*code*> the ending, of a section of text that is to be handled in ways specified by the particular code.
  - Pairs of beginning-and-ending codes can be inserted inside other such pairs.
  - Some codes have only a "beginning" value, such as a new-line code that tells the interpreter to begin a new line in the print-out or display at the point where that code occurs.
  - Text that has been marked up is typically processed by a computer program that interprets the codes and prints or displays the text in the ways that the codes direct.

Markup Languages

- A subset of SGML that has become familiar to many people through its use in the World-Wide Web is Hypertext Markup Language (HTML).
- Text that has been marked up with HTML is typically interpreted by a Web browser (e.g., Internet Explorer, Netscape) and displayed on a monitor. If the user directs, the browser will also print out the interpreted text.
- Text can be marked up with HTML codes (indeed, with codes in any SGML) using ordinary word-processing programs. This is an important strength of SGML.
  – However, markup in HTML is typically accomplished nowadays by special-purpose programs such as Microsoft FrontPage and Macromedia Dreamweaver.
  – Such programs greatly assist a human by inserting most of the needed HTML codes automatically and by immediately displaying the result on the monitor in WYSIWYG (What You See Is What You Get) form.

Markup Languages

- Examples of HTML codes and their meanings
  - <html> and </html> Begin and end use of HTML coding
  - <title> and </title> Begin and end title
  - <p> and </p> Begin and end paragraph
  - <h2> and </h2> Begin and end 2nd-level header
  - <em> and </em> Begin and end emphasis (depending on various settings, the browser may interpret this by using either bold type or italic type)
  - <br> Begin new line (i.e., branch) at this point in text
  - <hr> Insert horizontal rule at this point in text
  - <a href="http:*hyperlink*"> and </a> Begin and end link to a specified Webpage (i.e., a hyperlink)
  - <a href="mailto:*address*"> and </a> Begin and end an instruction that opens an email program associated with the browser and prepares it to send a message to the specified email address

Markup Languages

- An important superset of HTML is another subset of SGML known as Extensible Markup Language (XML).
  - XML is becoming widely used on the World-Wide Web and on the Internet in general as a way of extending the markup of text to include such capabilities as identifying types of elements in complex structures such a databases.
    - For example, in XML a user can define codes that will mark the Social Security Number field, the employee-surname field, and the employee-salary field in a database.
  - A major strength of XML is that, unlike HTML (which uses codes standardized via the ISO), new codes in XML can be freely defined for use in any particular application, provided only that the users of that application agree on the meaning of the specific codes.
    - This important feature is why XML is extensible.

Markup Languages

- Still another example of an SGML is SMIL, Synchronized Multimedia Integration Language,
  - SMIL lets users define and manipulate multimedia elements (e.g., photographs and other still images, as well as audio and video recordings) in Webpages in a coordinated fashion (rather than only as independent elements, which is all that HTML can manage).
    - For example, a succession of diagrams can be presented at suitable intervals, each accompanied by a recorded voice commentary, with background music extending through the whole set of diagrams.
  - SMIL can be regarded as an example of an implementation of XML.

History of Applications Programming

- From the 1940s till the late 1970s an applications programs would be written for a particular type of computer.
- Further, even when a copy of the program was installed on one of the intended computers, the copy would have to be tailored by hand for the particular configuration of that computer, e.g., in terms of the specific identities of the tape drives used at that particular location.

History of Applications Programming

- Again, till the late 1970s there would be at most a few thousand installed units of any particular computer.
- A consequence of the relatively small numbers of copies of even the most successful programs, and of the hand tailoring needed for each copy, was high prices for software.
  - I remember a database-management (DBM) program that cost $100,000 per copy in 1969 (equivalent to $470,000 in today's dollars), plus hundreds of dollars per month for maintenance.
  - That DBM program was less powerful and far less convenient to use than today's Microsoft Access, which costs about $200 a copy.

Computer Program Ingredients

- Computer programs consist of a main sequence of instructions (sometimes called the ***kernel*** sequence), which the CPU follows from start to finish, unless one or more of certain circumstances occurs (as will almost certainly be the case).
- One circumstance is that in the kernel there may be logical or arithmetic ***tests*** of various conditions. The outcome of a test (also called a ***decision point***) may cause the CPU to ***jump*** to some other part of the kernel, and continue from there. (Jumps are also sometimes called ***branches***.)

Computer Program Ingredients (cont'd)

- Some jumps are to what are called ***subroutines***. Subroutines are subsets of instructions that carry out certain operations, often processing values passed to the subroutine at its start (e.g., values associated with the test that led to the jump)
- When a subroutine has finished, the CPU may return to the kernel (at the next instruction after the one that caused the jump), or the CPU may jump to another subroutine.
  - Which of these possibilities occurs depends upon the nature of the subroutines and upon tests that may be performed within the subroutine.
  - If the CPU jumps to another subroutine, then when it finishes there, it may jump back to the first subroutine (at the next instruction after the one from which it jumped), or the CPU may jump to still another subroutine. And so on.

Computer Program Ingredients (cont'd)

- Subroutines are a major tool in programming. They enable relatively small sets of instructions to be tested thoroughly and used over and over again.
- A given subroutine may be jumped to (or ***called***) from many different places in the kernel sequence of instructions in a program, or from many other subroutines.

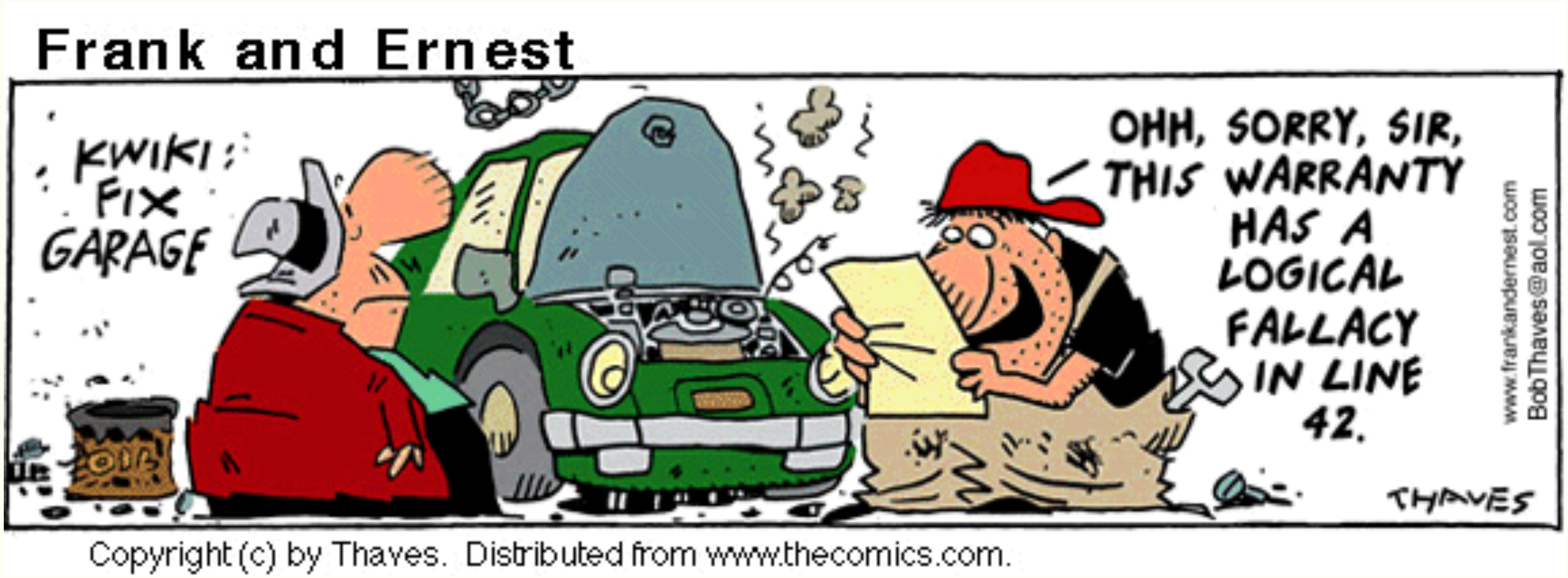
Computer Program Ingredients (cont'd)

- Another of the circumstances in which the CPU may fail to simply move linearly through the kernel is that there may be places in the kernel (or in other sequences, such as subroutines) where a certain subset of instructions is repeated either
  - For a specified number of times; this is called a ***counting loop***; or
  - While some condition persists, or till some condition is initiated; these are called ***do-while loops*** and ***do-until loops***.

Computer Program Ingredients (cont'd)

- In short, the principal computer-program ingredients are
  - Main sequence of instructions, or kernel
  - Logical or arithmetic tests, or decision points, that can lead to jumps or branches
  - Loops: viz., counting, do-while, and do-until
  - Subroutines
- Out of these seemingly simple ingredients comes the enormous variety of things that computer programs can do.

Computer Program Ingredients (cont'd)


Frank and Ernest
KWIKI FIX GARAGE
OHH, SORRY, SIR, THIS WARRANTY HAS A LOGICAL FALLACY IN LINE 42.
THAVES
www.frankanderenest.com
BobThaves@aol.com
Copyright (c) by Thaves. Distributed from www.thecomics.com.

It is a pleasure to acknowledge with gratitude that the above strip is used here with the kind permission of Bob Thaves.

Computers and Programs

- What can we say about the future of computers and computer programs?
- I think there are three sure predictions to be made:
  - There is good reason to believe that hardware speeds and software complexity will continue to increase indefinitely during the foreseeable future.
  - Culture and civilization, already irrevocably dependent on computers, will become still more dependent.
  - Within a few years computers will be doing things that today we have not yet even dreamed of their doing.
    - For example, in 1990 no one dreamed that there would be a World-Wide Web with the ability to carry full-speed television programs.

Computers and Programs

- Thinking about the future of computing, I cannot resist quoting again from F. P. Brooks's *The Mythical Man-Month*:
  - "Still vivid in my mind is the wonder and delight with which I—then 13 years old—read the account of the August 7, 1944, dedication of the Harvard Mark I computer, an electromechanical marvel for which Howard Aiken was the architect. . . . Equally wonder-provoking was the reading of Vannevar Bush's "[As] We May Think" paper in the [July] 1945 *Atlantic Monthly*, in which he proposed organizing knowledge as a big hypertext web and giving users machines for both following existing trails and blazing new trails of associations.
  - "My passion for computers got another strong boost in 1952, when a summer job at IBM in Endicott, New York, gave me hands-on experience in programming the IBM 604 and formal instruction in programming IBM's 701, its first stored-program machine. Graduate school under Aiken and Iverson at Harvard made my career dream a reality, and I was hooked for life. To only a fraction of the human race does God give the privilege of earning one's bread doing what one would have gladly pursued free, for passion. I am very thankful.

Computers and Programs

– "It is hard to imagine a more exciting time to have lived as a computer devotee. From mechanisms to vacuum tubes to transistors to integrated circuits, the technology has exploded. The first computer on which I worked, fresh out of Harvard, was the IBM 7030 Stretch supercomputer. Stretch reigned as the world's fastest computer from 1961 to 1964; nine copies were delivered. My Macintosh Powerbook is today not only faster, with a larger memory and bigger disk, it is a thousand times cheaper. (***Five*** thousand times cheaper in constant dollars.) We have seen in turn the computer revolution, the electronic computer revolution, the minicomputer revolution, and the microcomputer revolution, each bringing orders-of-magnitude more computers.

Computers and Programs

– "The computer-related intellectual discipline has exploded as has the technology. When I was a graduate student in the mid-1950s, I could read ***all*** the journals and conference proceedings; I could stay current in ***all*** the discipline. Today my intellectual life has seen me regretfully kissing subdiscipline interests good-bye one by one, as my portfolio has continuously overflowed beyond mastery. Too many interests, too many exciting opportunities for learning, research, and thought. What a marvelous predicament! Not only is the end not in sight, the place is not slackening. We have many future joys."

The quotation on this and the preceding two slides is from: Brooks, Frederick P., Jr. (1995). *The Mythical Man-Month: Essays on Software Engineering: 20th Anniversary Edition*. Reading, MA: Addison-Wesley. ISBN:0-201-83595-9. [This edition is a re-issue, with additional material, of the initial publication in 1975.]

The Sky's the Limit!