

globally distributed manufacturing is possible. Virtual corporations can be established involving participants from around the world. Design, production, and assembly can occur wherever is most convenient and cost-effective for the enterprise. Developments in CAD/CAM, including advances in product data models and international standards, have made this possible. See COMPUTER PROGRAMMING; DIGITAL COMPUTER; ENGINEERING DESIGN; INTERNET; WORLD WIDE WEB. K. Preston White, Jr.; Larry G. Richards

Bibliography. C. R. Asfahl, *Robots and Manufacturing Automation*, 2d ed., 1992; D. D. Bedworth, M. R. Henderson, and P. M. Wolfe, *Computer-Integrated Design and Manufacturing*, 1991; T.-C. Chang, R. A. Wisk, and H.-P. Wang, *Computer Aided Manufacturing*, 2d ed., 1998; C. Machover, *The CAD/CAM Handbook*, 1996; C. McMahon and J. Browne, *CAD/CAM: Principles, Practice and Manufacturing Management*, 2d ed., 1998; P. K. Wright and D. A. Bourne, *Manufacturing Intelligence*, 1988; I. Zeid, *CAD/CAM: Theory and Practice*, 1991.

Computer-aided engineering

Any use of computer software to solve engineering problems. With the improvement of graphics displays, engineering workstations, and graphics standards, computer-aided engineering (CAE) has come to mean the computer solution of engineering problems with the assistance of interactive computer graphics. See COMPUTER GRAPHICS.

CAE software is used on various types of computers, such as mainframes and superminis, grid-based computers, engineering workstations, and even personal computers. The choice of a computer system is frequently dictated by the computing power required for the CAE application or the desired level and speed of graphics interaction. The trend is toward more use of engineering workstations. See DIGITAL COMPUTER; MICROCOMPUTER.

Design engineers use a variety of CAE tools, including large, general-purpose commercial programs and many specialized programs written in-house or elsewhere in the industry. Solution of a single engineering problem frequently requires the application of several CAE tools. See ENGINEERING DESIGN.

A typical CAE program is made up of a number of mathematical models encoded by algorithms written in a programming language. The natural phenomena being analyzed are represented by an engineering model. The physical configuration is described by a geometric model. The results, together with the geometry, are made visible via a user interface on the display device and a rendering model (graphics image). See ALGORITHM; COMPUTER PROGRAMMING; MODEL THEORY; PROGRAMMING LANGUAGES.

CAE allows for many more iterations of the analysis-design cycle than was possible by hand computation, especially when the CAE is coupled with optimization systems that drive this cycle automatically. The benefits are translated into improved productivity and quality of design.

Program structure. A CAE program usually consists of a series of mathematical models and a data structure. **Figure 1** illustrates a simplified view of a typical CAE program operating in an engineering workstation environment. First, a mathematical description of the physical phenomena being analyzed is written. This engineering model may consist of equations such as Newton's second law to describe the dynamics of a system or the Navier-Stokes equations to analyze a fluid flow field. Next, a model of the physical configuration is created. This geometric model may consist of two- or three-dimensional (2D or 3D) curves, surfaces, faceted approximations to surfaces, or solid elements. The results of the engineering analysis are frequently displayed on the geometric model by color fringing to show the variation of a scalar parameter. Large amounts of data are created during this modeling phase, and the need for a data structure to store and retrieve them is greater than for the engineering model. See DATA STRUCTURE.

Although the engineering and the geometry are fully described, they cannot be viewed on the display until a model for rendering has been formulated and coded. A mathematical description of the lighting conditions, the approximate intensities of

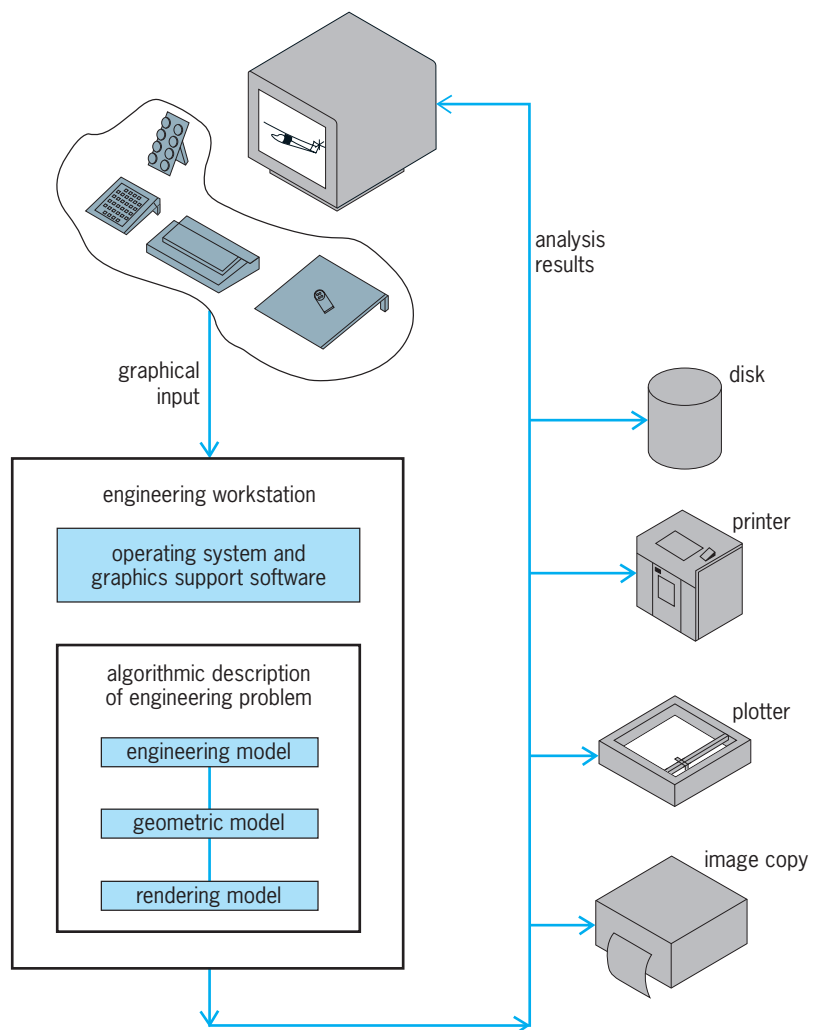


Fig. 1. CAE program in an engineering workstation environment.

light reflected by the nodes of the geometric model, and the corresponding shades of color provide the graphics data necessary for the model to be realistically shaded. Before viewing, the graphics data are transformed from geometric model coordinates to a normalized coordinate system. Parameters are set for an orthographic or perspective projection from 3D model coordinates to 2D coordinates for final transformation to the display screen in device coordinates. The CAE applications programmer accomplishes most of these tasks with the help of precoded algorithms provided by graphics-support software. Interactive communication with the graphics image and the geometric and engineering models occurs through a user interface written with the assistance of the graphics software or windowing software provided with the workstation.

In the past, CAE programs have been predominantly coded in the FORTRAN programming language. The present trend is toward the C, C++, and Java programming languages together with the UNIX and Windows operating systems. See OPERATING SYSTEM.

Graphics standards. Portability, the ability to move programs easily from one computer to another, is important for CAE software. Although it has long been possible to make a computational program code portable by using standard programming languages, this was not previously possible for CAE software because of the lack of graphics standards. A proposed 3D graphics standard (CORE), introduced to the American National Standards Institute (ANSI), was superseded by the adoption in 1985 of the 2D Graphical Kernel System (GKS) as an international standard by the International Organization for Standardization (ISO).

A new 3D device-independent graphics standard, the Programmer's Hierarchical Interactive Graphics System (PHIGS), was proposed by ANSI in 1985 and adopted as an international standard by ISO in 1988. Where possible, the concepts and nomenclature of GKS have been used in PHIGS.

The principal limitation to CAE software portability has been the wide variety of graphics hardware and the direct dependence of CAE software on this hardware. Both GKS and PHIGS give the programmer device-independent graphics primitives and coordinate systems as well as a set of logical graphics input devices to replace the wide variety of input hardware. For example, the pick action (selection of a graphics entity on the screen) could be physically accomplished by a light pen, a cursor and tablet, or a mouse. Using graphics standards, the CAE programmer will always specify a logical pick device regardless of the physical device used to achieve the pick.

PHIGS has several important advantages over GKS for CAE software. It is a full 3D system for viewing and modeling transformations, and allows both graphics and nongraphics data to be stored in its data structure. The data structure can invoke other structures and store transformations as attributes.

The result is a hierarchical graphics data structure well suited for animation and representation of entities with multiple components. A number of extensions to PHIGS, called PHIGS PLUS (PHIGS+), were adopted as a separate international standard in 1990. PHIGS+ provides support for most of the rendering model and some of the geometric model in a CAE program. PHIGS+ routines address lighting, shading, hidden surface elimination, transparency, nonuniform rational B-spline (NURBS) curves and surfaces, and improved user interaction and control. PHIGS and PHIGS+ were merged into a single international standard, PHIGS, in 1997.

The emergence of PHIGS+ was in response to dramatically improved computer graphics hardware capabilities in the late 1980s and early 1990s, and that were rapidly being incorporated into the new breed of CAE workstations. Vendors did not wait for international committees to address these new capabilities in their standards, but deployed their own proposed extensions in the interim. While this made these new capabilities available to the CAE software developers, it also prevented the CAE software from easily being ported from one hardware platform to another, which is the main purpose of a standard.

Taking advantage of this noncompliance to the adopted international standard and the slow rate at which new technologies and capabilities were addressed by these standards, an industry consortium was established in 1992 to support a new graphics standard known as OpenGL. The aim of OpenGL is to provide a timely integration of new technologies and capabilities as they emerge and to avoid the problem of partial conformance to the standard. GKS, PHIGS, and other standards have been plagued by partially conforming deployments, which makes portability between platforms difficult and defeats a key purpose of standards. The OpenGL Architecture Review Board, which draws its membership from nine leading computer graphics hardware vendors, avoids this problem by permitting only the term OpenGL to be applied to products that conform completely to the OpenGL standard. OpenGL is today the most widely used and supported 2D and 3D graphics application programming interface (API).

Hardware. Computer systems ranging from workstations to mainframe computers are used for CAE software. For example, the aerospace and automotive industries make wide use of very large mainframe computers and large grid-based computer systems to support computer-intensive CAE software such as finite element analysis and computational fluid dynamics. These large computational resources are connected by high-speed data transmission lines to file servers holding the model and resulting data. The precomputation model description and the postcomputation interrogation of the results, and frequently the computations themselves, are usually performed on ultra-high-end workstations capable of managing these vast amounts of data. These workstations are typically 64-bit multiprocessor systems, with 12 to 24 gigabytes (GB) of shared memory

(RAM) and a UNIX operating system, which are connected to the file servers via one or more 1- or 10-gigabit/second network connections. Ultra-high-end workstations are necessary when working with large, complex models and datasets, such as the solid modeling of a complete automobile or aircraft.

For less complex models and datasets, or for subsets of the larger models and datasets, it is typical to use high-end engineering workstations. These workstations are typically 32-bit single- or dual-processor systems with up to 4 GB RAM running a Windows or UNIX operating system, the former being more common. These engineering workstations distinguish themselves from personal computers by having more RAM and faster central processing units (CPU), graphical processing units (GPU), hard disks, network connections, and internal interconnects and data transfer rates. These high-end engineering workstations cost two or three times more than a typical personal computer and are commonly used by engineers. In contrast, ultra-high-end workstations typically cost three to five times more than a high-end engineering workstation, and consequently are less used.

Computer-intensive engineering problems are frequently distributed across multiple CPUs. These CPUs can be connected in a grid structure where each CPU has its own RAM and where communications between these CPUs and their RAM takes place over a computer network. Occasionally these grids are composed of regular personal computers and engineering workstations that are accessed whenever they are idle, such as during nighttime hours. The bottlenecks of such systems are the limited bandwidth between the CPUs and the personnel time required to configure and manage these grids. Grid systems tend to be most successful when the computations can be easily parallelized and when there is minimal need to communicate and move data between the CPUs. Computational problems that do not fall into this category are best performed on large multiprocessor, shared-memory systems with high-speed, high-bandwidth interconnects between the CPUs. Examples of such systems include mainframes and the ultra-high-end workstations described earlier. The advantages of these systems include the flexibility of how data can be accessed simultaneously by multiple CPUs without having to be moved across a network, and the reduced need for manually managing the distribution of the computational resources within the system. *See* MICROPROCESSOR; MULTIACCESS COMPUTER; MULTIPROCESSING; SUPER-COMPUTER.

These large centralized computational resources are often connected to networks of from 10 to 100 client workstations that act as display terminals with local controllers (Fig. 2). When a large number of CAE users invoke computer-intensive programs, a significant delay in response usually occurs. For this reason, as well as initial cost considerations, most companies today deploy high-end engineering workstations on a two- or three-year upgrade cycle.

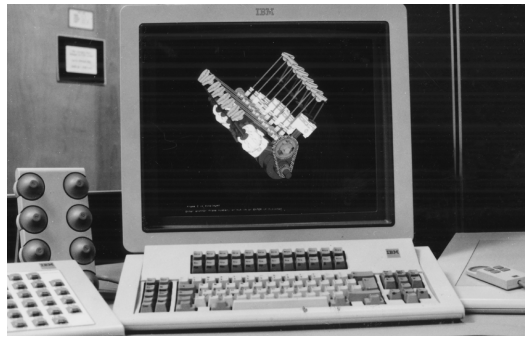


Fig. 2. Solid modeling and simulation on a mainframe via a CAE workstation.

Though many small CAE applications programs are written for personal computers, of which most engineering companies have a wide array, most large, commercial CAE programs do not perform effectively or at all on personal computers. Engineering workstations, on the other hand, have the necessary computer power and local resources, and usually run only one CAE program at a time and serve only one user. As a result of this independence, a demanding CAE code at one workstation node in a computer network will not affect the response time for other users. *See* DISTRIBUTED SYSTEMS (COMPUTERS); LOCAL-AREA NETWORKS.

The personal-computer gaming industry has been instrumental to the recent widespread availability of affordable, high-quality graphical processing units that are capable of supporting the heavy demands of CAE graphics processing. These graphical processing units contain a number of specialized high-speed, parallel graphics engines, display list memory, image memory, buffer memory, and alpha memory to permit the real-time generation of color-shaded images desired for CAE applications. The graphics engines perform graphics modeling and viewing transformations, lighting and shading computations, and scan conversion of models for raster-scan display. Display list memory retains the rendering model structure. The z-buffer provides fast hardware support for hidden surface elimination, whereas alpha memory allows fast hardware-assisted transparency computations. The final rasterized image is stored in one of two image memories for fast transmission to the display screen. Most graphical processing units support the OpenGL graphics standard by performing its operations directly in hardware as opposed to computing them in software. *See* COMPUTER ARCHITECTURE.

The graphical processing unit used in a typical personal computer accounts for approximately 2–4% of the personal computer cost. In contrast, in a high-end engineering workstation, the graphical processing unit typically accounts for 20% of the hardware cost. Where ultra-high-end workstations are also used for graphics processing and visualization, it is increasingly common to use multiple high-end graphical processing units working in parallel, with each unit capable of computing 750 million triangles per second or 8 billion pixels per second.

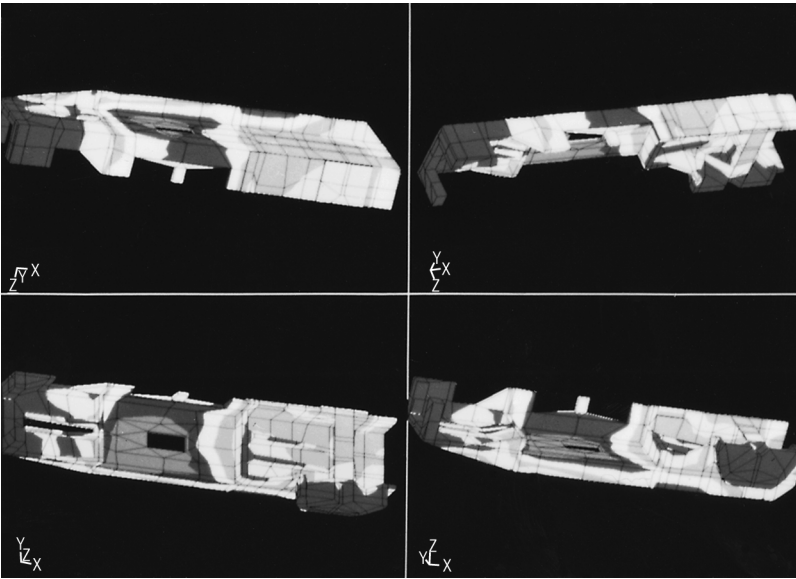


Fig. 3. Finite-element stress analysis using commercial CAE software.

A wide variety of output devices are available for CAE images. The most popular are ink-jet and laser printers for color images and line drawings on paper and Mylar. Others include wall projection and stereovision display systems, often combined with positional and orientation input from the observer. Many types of graphics input devices are available for workstations and mainframe displays. Cursor and tablet, mouse, joystick, ball, glove, function key box, and dial box are the most widely used. Light pens are generally in use on older-technology displays. See COMPUTER PERIPHERAL DEVICES.

Commercial CAE software. Most companies use a combination of a few commercial CAE programs and a number of smaller, specialized CAE applications programs typically written in-house or elsewhere in the industry. There are several classes of commercial CAE software. The first large commercial programs used for CAE were dynamic simulation systems, which usually had no graphical output or at best a printer-plotter output of graphs. They required the engineer to give input in the form of ordinary differential equations or building-block elements to describe the dynamic behavior of a real system, in many ways similar to the patching (programming) of analog computers. These systems numerically integrate the set of coupled differential equations, thereby simulating the dynamic behavior of the particular system. Simulation systems are still widely used, and although the graphics output has improved considerably, the description of the input has not changed greatly. See ANALOG COMPUTER; NUMERICAL ANALYSIS; SIMULATION.

A need to solve detailed problems, such as stress or deformation analysis, for components of systems led to the development of the finite-element method or finite-element analysis. In this method, a complex object is broken down into simpler elements. With these, a set of equations is formulated which, when solved, predict the behavior of the object as modeled by the set of elements. The modeling of the object is

known as finite-element preprocessing, and the validity of the solution is largely dependent on the skill of the modeling engineer. Although very advanced methods of interactive graphics are used in finite-element preprocessing, it is still a very tedious and expensive operation. The solution of the equations is highly computer-intensive. Finite-element methods are applicable for a wide variety of physical phenomena, including mechanical stress and strain, fluid flow, acoustics, heat transfer, and electrical fields.

Figure 3 illustrates the solution of a finite-element stress analysis problem with a commercial CAE code running on a mainframe computer. See FINITE ELEMENT METHOD.

Computer-aided design and manufacturing. CAD/CAM systems were created by the aerospace industry in the early 1960s to assist with the massive design and documentation tasks associated with producing airplanes. By the late 1970s, these codes were being distributed to other industries. CAD/CAM systems have been used primarily for detail design and drafting along with the generation of numerical control instructions for manufacturing. Gradually, more CAE functions are being added to CAD/CAM systems. A trend toward open architecture with flexible geometry interfaces is stimulating the addition of more analysis and manufacturing functions. Modeling with CAD/CAM systems has become fairly sophisticated. Most popular commercial systems support 2D and 3D wireframe, surface models, and solid models. Rendered surface models differ from solid models in that the latter have full information about the interior of the object. For solid models, a combination of three types of representation is commonly used: constructive solid geometry, boundary representation, and sweep representation. Complex systems require significant amounts of processing power, primary memory space, and disk space. See COMPUTER-AIDED DESIGN AND MANUFACTURING.

Integration of CAE software. Frequently, all of these CAE tools are needed together with specialized CAE applications programs to solve a single engineering problem. The integration of these tools or the communication of data between them is a challenging problem. To successfully integrate these programs, a centralized database is necessary. Data from the various CAE tools are processed into and retrieved from this database through a database management system. If the CAE software is proprietary and the data structure not easily accessible, the proprietary programming interfaces may be used or, sacrificing of functionality and interactivity, a data interchange standard may be used. See DATABASE MANAGEMENT SYSTEM.

The Initial Graphics Exchange Specification (IGES) was developed under the leadership of the National Bureau of Standards (today known as the National Institute of Science and Technology) and was accepted as a standard by ANSI in 1981 (Y14.26M-1981). The goal of IGES is to allow the transfer of geometric data between dissimilar CAD systems. IGES is widely available and is generally capable of providing the translation of a snapshot of the model in one CAD

system into a static model that cannot be edited in another CAD system.

The standard for exchange of product model data (STEP) has been adopted as an international standard by the International Organization for Standardization (ISO 10303). The aim of STEP is to provide a neutral, computer-interpretable representation and description of product data throughout the life cycle of a product that is independent from any particular system. It can be used for both data exchange and for archiving data over time. The latter is an important issue when the product life cycle exceeds the life span of the software and hardware that created the product data. Given this extremely broad objective, ISO 10303 is not a single standard but a collection of interrelated documents that form a multipart standard. A number of these documents have been adopted as an international standard, while many others are still in development. The **table** lists of some of the documents that have been adopted as part of ISO 10303.

The software industry's support for adopted ISO 10303 documents has been significantly slower than was the case for PHIGS (graphics standards). In the case of STEP, the software vendors have been hesitant to support the standard, both because of the high up-front cost associated with complying with the complexity of the standard, and because of their desire to provide a comprehensive yet proprietary solution for the customer in which the customer's data is locked into. Thus far, it is primarily the large mechanical CAD/CAM/CAE vendors serving the automotive industry that have given into customer pressure and competition to provide support for STEP (AP 203, AP 214). Recently it appears that AP 210 and AP 212 have also started to receive some initial vendor support.

Adopted ISO 10303 documents

Document number	Title
STEP AP 201	Explicit Draughting
STEP AP 202	Associated Draughting
STEP AP 203	Configuration Controlled Design
STEP AP 204	Mechanical Design using Boundary Representation
STEP AP 207	Sheet Metal Die Planning & Design
STEP AP 209	Composite and Metallic Structural Analysis and Related Design
STEP AP 210	Electronic Assembly, Interconnection and Packaging Design
STEP AP 212	Electrotechnical Design and Installation
STEP AP 214	Core Data For Automotive Mechanical Design Processes
STEP AP 215	Ship Arrangement
STEP AP 216	Ship Moulded Forms
STEP AP 218	Ship Structures
STEP AP 224	Process Planning Using Machining Features
STEP AP 225	Building Elements Using Shape Rep
STEP AP 227	Plant Spatial Configuration
STEP AP 232	Technical Data Packaging: Core Info & Exch.

Applications. The CAE methods for electrical and electronics engineering are well developed. The geometry is generally two-dimensional, and the problems are primarily linear or can be linearized with sufficient accuracy. Chemical engineering makes extensive use of CAE with process simulation and control software. The fields of civil, architectural, and construction engineering have CAE interests similar to mechanical CAE with emphasis on structures. Aerospace, mechanical, industrial, and manufacturing engineering all make use of mechanical CAE software together with specialized software.

An example of CAE is the design of an aircraft landing-gear mechanism. The first step is definition of the problem and creation of a set of performance specifications. Next, the conceptual design phase may be aided by specialized programs to determine a size estimate of the landing gear based on specified loads and deflections. Commercial CAE programs are available for kinematic synthesis of mechanisms based on specified motion requirements, but this landing-gear mechanism will be designed with an in-house program written for the purpose. *See LANDING GEAR.*

The next phase is preliminary design. An applications program is used to analyze the deflection and response of a shock-absorbing, energy-dissipating strut. Dynamic analysis of the guiding mechanism and complete assembly is determined by a commercial code. When the dynamic loads are determined, a finite-element stress analysis of each link of the mechanism is done by using commercial finite-element-method software. Following the stress analysis, some links are changed in size and the dynamic analysis repeated to determine new loads. Using the new loading, another iteration of the finite-element-method software is made to verify that the stresses fall below the strength limits.

The next phase is the final design. All components of the assembly are drawn in 2D on a CAD/CAM system and detailed, giving dimensions, material specifications, and other instructions. An assembly drawing is created from the components, and mating of components are verified. An alternative approach is to create a solid model of each component, assemble the solid components, and run an automatic interference-clearance check; 2D drop-offs are then automatically made of each component and manually detailed (at the workstation) by a drafter. From the final part geometry, instructions for numerically controlled machine tools are generated to produce the part. Some systems may support tooling design and process planning. Finally, a design release is made to the manufacturing department.

Jan Helge Bøhn; Arvid Myklebust

Bibliography. J. D. Foley et al., *Introduction to Computer Graphics*, 1993; T. Gaskins, *PHIGS Programming Manual*, 1992; M. K. Gillenson, *Database, Step-by-Step*, 2d ed., 1990; F. R. A. Hopgood et al., *Introduction to the Graphical Kernel System (GKS)*, 2d ed., 1987; M. E. Mortenson, *Geometric Modeling*, 2d ed., 1997; D. F. Rogers, *Procedural Elements for Computer Graphics*, 2d ed.,

1997; D. Shreiner et al., *OpenGL Programming Guide*, 5th ed., 2005; A. Tizzard, *An Introduction to Computer-Aided Engineering*, 1994.

Computer architecture

The art or practice of designing computer systems. Just as the architecture of a building describes its overall structural concept and principal features, the architecture of a computer system consists of a description of the overall layout and major features of a computer system. In both cases, we are concerned with how the object, whose architecture we are describing, appears to the user. In computer architecture, the user is not the final user of the system, but the person who writes computer programs to be run on the system. Compared to the architecture of buildings, which is largely an issue of esthetics, the architecture of a computer system is principally about efficiency and economy of operation. There may be an esthetic dimension to computer architecture, but when present it is distinctly lower in importance.

At present, computer architecture is the highest level of the closely related field of computer design. The field has evolved from being the top level of the design process used by the early developers of computers into a separate area of specialization within the overall hierarchy of computer designers and engineers. Underneath computer architecture are the fields of computer organization and implementation, and computer engineering. *See* COMPUTER; DIGITAL COMPUTER.

Computer organization and implementation. Closely related to the field of computer architecture is a subsidiary discipline known as computer organization and implementation. It is difficult to define precisely where computer architecture stops and computer organization and implementation begins. In general, computer architecture is concerned with the principal design parameters, while computer organization and implementation is concerned with the subsidiary points of design of the system, including the composition of the logic circuits that make up each structural component of the system and the implementation of each of those logic circuits as an arrangement of specific electrical circuits of some particular type or types. Computer organization focuses on issues such as whether an adder circuit will be implemented as a ripple-carry adder or as a carry-lookahead adder and whether the underlying electrical circuits will be implemented in TTL (transistor-transistor logic), bipolar transistors, CMOS (complementary metal-oxide semiconductors), or gallium arsenide (GaAs). *See* INTEGRATED CIRCUITS; LOGIC CIRCUITS.

Major issues. There are several major issues in the design of a computer system covering all or nearly all of the architectural considerations. *See* MICROCOMPUTER.

Instruction-set architecture. Traditionally, the instruction-set architecture—the detailed design of the set of machine instructions implemented in the

processor—has been the core issue in defining the architecture of a computer system. For example, how many machine instructions will there be, will the instructions be of uniform length or variable in length, and how many fields will the machine instruction contain. *See* MICROPROCESSOR.

Registers. The typical central processor contains several special-purpose registers (storage units), including an instruction register, a program counter or instruction pointer, a processor status register, perhaps one or more separate index registers, segment (base) registers, one or more general-purpose registers to hold integer or fixed-point data (integer registers), and either the same or a distinct set of registers to hold floating-point data. Although integer registers, a single instruction register, and the program counter or instruction pointer have traditionally been of the same width (number of bits), there is no law requiring such. Floating-point registers may or may not be of different width from integer registers. *See* COMPUTER STORAGE TECHNOLOGY.

Native data types. These are the data types that will be separately and distinctly represented and recognized by the computer hardware. Some computers may natively represent and carry out arithmetic on decimal integers via an underlying binary representation. Floating-point representation may or may not be present. *See* NUMBERING SYSTEMS.

Word size. The most commonly encountered number of bits contained in a single word of memory (word width) are 32 and 64. *See* SEMICONDUCTOR MEMORIES.

Memory addressing scheme. There are two principal issues under the topic of memory addressing scheme. The first is the smallest unit of addressable memory, which is usually the byte, although there are several machines of major significance that are not byte-addressable but word-addressable. The second issue is the order in which the bytes are stored for the several bytes that constitute a single word. In the scheme known as big-endian, the leftmost byte (that is, the byte with the numerically smallest address) contains the most-significant bits of the word content, while the byte containing the bits of least significance possesses the numerically highest address. The opposite arrangement is known as little-endian.

Maximum size of memory. A major issue for the modern computer architect is how large the memory address space should be, since memory has been getting cheaper at a prodigious rate. For every doubling of size of the memory address space, the width of an address increases by one bit. Therefore, the designer is not free to increase the memory address space by an arbitrary amount, as every increase in address width has much wider implications for instruction set architecture, as well as imposing requirements affecting lower levels of computer organization and implementation. On the other hand, insufficiency of memory address space has been one of the principal factors causing an architecture to become outmoded.

Number of buses and their relationship. A modern computer typically has a frontside bus and a backside bus,