

MICROPROCESSORS AND MICROCOMPUTERS

For articles on related subjects see **COMPUTER CIRCUITRY**; **EMBEDDED SYSTEM**; **INTEGRATED CIRCUITRY**; **INSTRUCTION-LEVEL PARALLELISM**; **MICROCOMPUTER CHIP**; and **REDUCED INSTRUCTION SET COMPUTER**.

A *microprocessor* is a single-chip integrated circuit (IC) implementation of a general-purpose central processing unit (CPU—*q.v.*). It contains a controller to direct the execution of program instructions, registers to store control and data values temporarily, and an arithmetic logic unit (ALU—*q.v.*) to calculate results. A microprocessor chip is a (very) large-scale integrated (LSI or VLSI) circuit fabricated on a sliver of silicon less than 7 mm (about 1/4 in) square and 0.5 mm (1/50 in) thick, about the size of a baby's fingernail.

In thirty years, microprocessors have gone from being electronic curiosities capable of doing simple arithmetic to being the fastest single-CPU computers on earth. They are becoming brain cells in a planet-wide computing network linking all human knowledge and making it widely and rapidly available everywhere (see **CYBERSPACE**). Microprocessors are the enabling engines of the information revolution presently sweeping technological society.

There are many different kinds of microprocessors, designed for many different applications. In 1990, there were about 50 different microprocessor families, including 995 varieties of CPU chips; in 1999, there were 34,000 CPU and controller chip types sold by 200 vendors. Each CPU chip contains the equivalent of 4,000 to 20,000,000 transistors; and chip complexity increases roughly by a factor of 100 every decade. A transistor-equivalent is able to make one simple logic decision or to store one binary digit (a 0 or 1 bit). A popular CPU categorization is based on the width in bits of its internal data path, commonly 16-bit, 32-bit, or 64-bit.

Processor architecture, such as the DEC Alpha series and the Sun Sparc family, is sometimes distinguished from a particular implementation. Architectures specify programmer-visible functions. A flexible architecture takes advantage of technological progress that allows ever faster, more highly integrated microprocessors.

A *microcomputer* is a more complete system than a microprocessor, containing not only the CPU logic, but also memory for storing programs and data plus input-output (I/O) interfaces for exchanging data with peripheral devices. In 1990, most microcomputers combined a CPU chip with 1 to 200 support chips on a single circuit board no larger than a double book page. Most extra chips contained from 262,144 (2^{18}) to 4,194,304 (2^{22}) bits of memory. A 100-chip micro-

computer usually controlled a workstation (*q.v.*) with I/O interfaces for a graphics display, mouse (*q.v.*) input keyboard, storage disk, loudspeaker, and network communications; it executed from 64 to 300 different instructions at rates from 1 million to 80 million instructions per second (MIPS) and contained 9 to 36 million bytes (megabytes, or MB) of memory, including its display image. A memory byte is eight bits and can hold one alphabetic character, two decimal digits, eight display dots, one color value, or part of a large binary number.

By 1999, the circuitry for a personal computer had shrunk to a module the size of a deck of playing cards and containing about 10 chips: CPU, memory bus, memory controller, I/O bus, I/O controller, and 2 to 8 memory chips of 67,108,864 (2^{26}) to 1,073,741,824 (2^{30}) bits each, giving 64 MB to 1,024 MB of memory. Common execution rates were 100 to 1,000 MIPS.

The ever-increasing number of transistors that a single integrated circuit can hold has made possible *single-chip microcomputers* (Fig. 1), small systems with large application markets. They range in processing speeds from slow, tiny appliance controllers to near-supercomputers (*q.v.*). A *digital signal processor* (DSP) is a microcomputer designed for high throughput (*q.v.*) numerical processing ("number crunching"). DSP

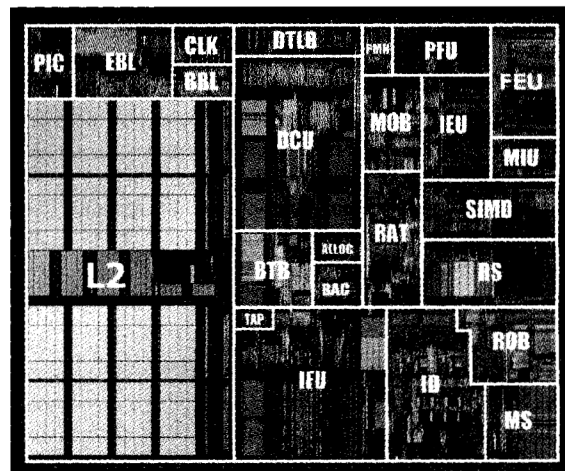


Figure 1. Photomicrograph of an Intel Pentium III processor introduced in 1999. It is a VLSI circuit with 28.1 million transistors. Among the major parts of the chip, marked with an overlay, are—left side: the cache memory (L2), and interface to the external bus (EBL); top center: data allocation units, including the data cache (DCU); bottom center: the instruction fetch unit and front-end of the instruction pipeline (IFU); top right: arithmetic execution units (IEU: integer execution), (FEU: floating-point execution), (SIMD: MMX technology instructions); bottom right: instruction decoder (ID) and associated logic for instruction scheduling. (Courtesy of Intel Corporation.)

chips are very useful for speech recognition and synthesis (*q.v.*). Their fast operations on arrays of numbers find many scientific applications. A *coprocessor* is an older, optional function unit on a separate chip that can rapidly execute complex floating-point arithmetic instructions (*see* COMPUTER ARITHMETIC). As of 1999, the millions of transistors on each fast CPU chip included all arithmetic units and logic for many DSP instructions.

A *bit-sliced microprocessor* is an obsolete type of fast CPU formed by combining several function blocks, each a separate chip. One chip family could be used to create prototypes of many different processors. The term *bit-slicing* (*q.v.*) describes the blocks for handling data within the CPU; they are 1 to 32 bits wide, but can be connected in parallel to form processors with any desired word width.

In the 1990s, a new technology became popular for making fast processor prototypes: *programmable gate arrays* (PGAs—*see* COMPUTER CIRCUITRY). Each chip holds regular arrays of hundreds of thousands of AND and OR gates with memory selectable interconnections. Some arrays, called *complex programmable logic devices* (CPLDs), also contain small fast RAMs for data and may have specialized DSP logic. Processor prototypes built with these logic devices execute programs hundreds of times faster than software simulators.

A *microcontroller* is a single-chip microcomputer built to control embedded systems, such as appliances and traffic lights. It typically has on-chip permanent program memory, a small amount of data memory for temporary values, and an assorted selection of interfaces and peripheral units such as serial and parallel ports, timers, and analog-to-digital converters (*q.v.*). Embedded applications usually do not require high performance, but need identical chips in huge quantities at very low prices.

Organization

A (micro) computer is organized into the three major von Neumann stored-program (*q.v.*) subsystems: processor, memory, and input-output. This design has stood the test of time surprisingly well, kept alive by improving implementation technology, even though it has an inherent speed limitation in exchanging data between the CPU and memory. Newer architectures, such as massively parallel machines that intersperse CPUs and input-output devices among memory modules, have the potential for high performance, but are of limited use, having yet to achieve standardization.

The *central processing unit* (CPU) contains the control units, execution units, and data registers that perform the instructions of a computer program. It can fetch

instructions and data from the memory system, store results back into memory, and can exchange data with the input-output subsystem.

The *control unit* is in charge of decoding instructions and sequencing the actions of various functional units. It resolves conflicts between on-chip resources and responds to internal and external interrupts (*q.v.*). Control units are simple state machines (*see* SEQUENTIAL MACHINE), implemented directly in logic circuitry (“hard-wired”) or by state table lookup (“microprogrammed”—*see* MICROPROGRAMMING). Hard-wired control is faster, but needs more IC space, is less flexible, and requires a simple internal architecture. A *pipelined* control unit can handle distinct parts of several simple instructions at once, greatly increasing execution rates. Processors with many complex instructions or legacy modes to emulate older CPUs are microprogrammed.

A *bus interface unit* controls instruction and data transfers to and from the CPU (*see* BUS). Architectures with separate instruction and data buses need two units—one for instruction fetch and one for data read and write. Sophisticated bus interfaces improve performance by prefetching instructions, buffering memory writes, and predicting branch instruction destinations.

There can be many *execution units*, commonly including an *arithmetic-logic unit* (ALU—*q.v.*), floating-point unit (FPU), and special-purpose units to execute graphics or vector instructions rapidly. The ALU executes most instructions on integer data, such as add, subtract, shift, rotate, and bitwise logical operations. If present, an FPU has specialized hardware to perform complicated floating-point arithmetic operations. FPUs often have distinct addition, multiplication, and division units, and dedicated registers. Many fast CPUs designed after 1997 have an on-chip graphics function unit.

A *register* (*q.v.*) is a storage location within an execution unit capable of holding one number. It is often part of a register set—a small on-chip data array that is explicitly or implicitly addressable by program instructions. Registers provide storage with the fastest possible access time and are included in all major microprocessors.

Registers are classified by function. *General-purpose registers* have no predefined functions; they can be used equivalently by programs to hold temporary data values during computations. Instructions access registers directly as operand (*q.v.*) sources or result destinations. Because reading a value from a register is much faster than reading from a memory location, keeping needed values in registers greatly improves execution speed. High-performance architectures all

have dozens or hundreds of general-purpose registers. For fast execution, optimizing compilers (*q.v.*) must carefully allocate registers to values.

Dedicated registers play specific roles during execution of some instructions. For instance, the stack pointer in the Motorola 68000 is one of the general-purpose registers, but is implicitly modified by instructions that access the stack. Many architectures place the program counter and condition codes in dedicated registers. Some older CPUs used a single *accumulator* register to provide one operand for each arithmetic operation and to receive each result.

Floating-point registers are general-purpose registers accessible only by floating-point and data transfer instructions. They provide storage space for lengthy floating-point numbers, freeing the integer registers. Having them as separate registers simplifies control of instruction pipelines (*q.v.*). When the FPU is a coprocessor, putting floating-point registers on the coprocessor greatly speeds execution by not requiring slow off-chip data transfers. By 1999, high-end CPUs had all the floating-point and integer registers and logic on one chip.

To the CPU, the *memory* subsystem appears to be a large array of locations that can hold data values or program instructions. *Random access memory* (RAM) has become synonymous with “computer main memory.” (see MEMORY). “Random access” means that any location can be referenced in the same time and manner. RAM can be dynamic (DRAM) or static (SRAM). DRAM is denser and needs much less power; SRAM is 10 times faster, but more costly. The densities achieved by DRAMs have made multi-megabyte primary memories affordable.

Read-only memory (ROM) is a simple type of memory with contents that cannot be changed, even by loss of electrical power. The contents are programmed during manufacture and are unalterable afterwards. ROMs are used for very high-volume control applications where simplicity and low per-unit cost, gained from mass producing the program within the memory chip, are critical.

PROMs, EPROMs, EEPROMs, and flash memory are erasable nonvolatile memories. A programmable read-only memory (PROM—see READ-ONLY MEMORY) can be written only once via an irreversible process. Data in Erasable PROM (EPROM) are reset by exposure to intense ultraviolet light. Both Electrically Erasable PROM (EEPROM) and flash memory are alterable by a high voltage input. Nonvolatile memories are very useful because they can survive power losses, but they can be rewritten only very slowly and for a limited number of times. Flash memory needs only 12 volts for

erasure and writes much faster than EEPROM; it can directly replace RAM for some applications. Recent research suggests that IC memories with minute magnetizable regions can be as fast to use and as easy to produce lithographically as DRAMs.

High-performance microprocessors need fast memory subsystems not costing much more than the cheapest memory arrays. The most important methods to improve price-performance are *cache memory* (*q.v.*) and paged *virtual memory* (*q.v.*). Caches greatly lower the average access time of system memory; virtual memory increases capacity. Both are part of a general memory hierarchy (*q.v.*) that encompasses all storage, from registers to magnetic tape.

Memory hierarchies rely on temporal and spatial locality of code and data references in running programs: after a memory location is accessed, it and nearby locations will often be needed many times. A large, cheap, slow memory can be combined with a small, expensive, fast memory to get both low cost and fast access. Currently needed regions are copied into fast memory and unneeded locations are returned to slow memory.

Cache memories are very fast SRAMs accessed directly by the processor and filled with copies of data from slower but cheaper DRAMs. Mappings between official DRAM addresses and actual SRAM locations are maintained by a cache controller. Average access times for memory with caching are one-fifth those for DRAMs alone. Paged *virtual memory* uses a similar, selective-copy scheme to make a slow disk appear to be a very large capacity RAM. Multitasking systems also use virtual memory to give each process its own separate memory, apparently starting at location zero. The virtual memory mappings of RAM to disk locations are maintained by the *memory management unit* (MMU) (see MEMORY MANAGEMENT). Cache memory started to be placed on CPU chips in the mid-1980s, and by 1999, most fast microprocessors had large caches on the CPU chip.

The *input-output* (I/O—see INPUT-OUTPUT OPERATIONS) subsystem includes all interfaces that connect the CPU to external peripherals, usually via device controllers that translate general commands from the CPU into specific details. A CPU may exchange I/O commands and data via direct I/O using special instructions, memory-mapped I/O (*q.v.*) by accessing controller registers as if they were memory locations, or direct memory access (DMA) by letting a sophisticated interface control the memory bus and transfer large blocks of data without CPU help. There are many types of device controllers, reflecting the many application areas for microprocessors. Most themselves contain an inexpensive microprocessor.

The most common I/O units are serial and parallel interfaces, or *ports* (see PORT, I/O). *Serial interfaces* accept data, such as 8-bit bytes, convert them into one-bit-at-a-time serial form, and send them over an optical fiber or a wire pair. The I/O interfaces must generate light or voltage signals corresponding to the data bits. *Parallel interfaces* are similar, but send or receive eight or more bits of data simultaneously on multistranded wire cables. *Network interfaces*, such as those for Ethernet (*q.v.*) and ATM (*q.v.*), allow computers to exchange data over communication links. They convert data to and from the special formats used in the network. All network devices have addresses. Propagation delays are longer than within a single computer system (see DATA COMMUNICATIONS).

Programmable *timers* interrupt the processor after a delay specified as a number of fundamental microcomputer clock cycles. They are very useful in real-time systems for providing a constant time delay between actions. *Analog to digital* (A/D) and *digital to analog* (D/A) converters (*q.v.*) are often provided on single-chip microcontrollers to transform analog input signals into digital form for processing, or vice versa.

Direct memory access (DMA) controllers can autonomously and rapidly transfer large blocks of data. They free the CPU to execute other tasks; they run transparently, without interfering with the CPU, by using the memory bus only when not needed by the CPU, in a process called cycle-stealing (*q.v.*). Fast disk controllers transfer data in DMA blocks.

Disk controllers are common peripheral interface units and one of the earliest produced in single-chip form. They convert I/O read and write requests into the right sequence of seeks, delays, and data transfer operations. Some disk interfaces—those following the small computer system interface (SCSI) standard, for example—assume that an intelligent controller in the disk drive accepts high-level commands to access stored data.

Video controllers put the electronics for a video display into one chip. Typically, they repeatedly read image data from memory, convert the data to pixels and colors, and send it to the screen. They calculate data addresses, look up colors, position pixels, and generate timing signals for the monitor. Some video controllers are general-purpose programmable microcomputers that also offer a number of different drawing primitives.

Special controller circuits for speech synthesis, speech recognition, and three-dimensional (3D) graphics imaging were added to I/O chips and some CPUs in the late 1990s. The Intel Pentium/MMX processor of 1997 and many later chips included special parallel

floating-point instructions to speed calculations for 3D graphics images.

History

Microprocessors and microcomputers are products of a microelectronic revolution characterized by ever-shrinking costs and sizes for information processing devices. The history of microprocessors is tightly interwoven with the histories of integrated circuits and the semiconductor industry.

Microelectronics began at Bell Laboratories in 1947 with the development of small transistor amplifiers on the surface of semiconductor materials, such as pure germanium and silicon (sand and glass are mainly silicon dioxide). A transistor is hundreds of times smaller than a vacuum tube, which it replaced for building computers by the late 1950s. After learning how to use photographic masks to control diffusion of critical trace impurities into tiny regions of silicon, by the mid-1950s engineers could produce batches of hundreds of individual transistors from a single, thin wafer of silicon.

In late 1958, Texas Instruments created an integrated circuit of five connected transistors. In 1959, Fairchild produced a planar transistor, the first device using integrated circuit (IC) technology on silicon with layers of silicon dioxide for insulation and thin films of evaporated aluminum for connectors. In 1961, Fairchild marketed the first commercially available IC, a flip-flop (1-bit) memory circuit with four transistors and two resistors. Most other early ICs were single-logic gates. Hundreds of IC copies are photographically reproduced at once upon a silicon wafer; they are tested to eliminate defective copies, cut into individual circuits about 5–7 mm ($\frac{1}{5}$ – $\frac{1}{4}$ in) square, and mounted in standard plastic or ceramic packages. The maximum number of elements per silicon IC has doubled every one to two years since 1959—an exponential phenomenon known as Moore's law (see LAWS, COMPUTER).

In 1967, ex-Fairchild executives formed Intel (see NOYCE, ROBERT). In 1969, Intel began developing a general-purpose chip set for handheld calculators. They abstracted the control mechanism into a programmable unit—the Intel 4004 chip. The idea of a controller on a chip originated with Gilbert Hyatt, who had the first microprocessor prototype running in 1968.

THE INTEL 80x86 FAMILY

The microcomputer revolution started in 1971, when Intel sold the first microprocessor chip, its 4004 controller. It contained 2,124 transistors, addressed 256

8-bit instructions, and operated on 4-bit data, each enough for a decimal digit (0–9). It was followed soon by the 4040 and several processors for 8-bit data: 8008 and 8080. The 8008 (1972) was designed to control video display terminals and keyboards. The 8080 and faster 8080A (1974) provided more registers, a stack (*q.v.*) for calling subroutines, and rudimentary 16-bit arithmetic. They executed 8-bit instructions, operated on 8-bit and 16-bit data, and had eight limited-use registers: a stack pointer (SP), three pairs of 8-bit registers (B, C, D, E, H, L), and an arithmetic accumulator (A). A register pair formed a simple index register addressing 2^{16} (65,536) memory bytes. The 8080 and 8080A were the first widely used microprocessors, used mainly in embedded control applications and in hobbyist systems that preceded personal computers (PCs). Their descendants are the 80x86 chips in many millions of PCs.

Improvements continued in 1978 with the 8086 and 8088, internally identical chips for 16-bit operations. The 8088 retained a narrow 8-bit external data bus for compatibility. Both used segmentation to extend the memory address range to 1 MB (2^{20} bytes). The 80186 (1981) was an 8086 with on-chip peripheral circuits for embedded applications. Subsequent improvements

to 8086 performance were made through the 80286 (1982), 80386 (1985), and the 80486 (1989) chips.

Starting with the 80386, the architecture was radically upgraded to form a 32-bit microprocessor supporting a nonsegmented 4 GB (2^{32} bytes) address space, virtual memory, and many new instructions. Compatibility with previous versions was provided by retaining the same register structure and supporting a virtual 8086 mode. The 80486 had the same instructions as the 80386, but added an on-chip memory cache, instruction pipelines, and a floating-point unit to improve performance greatly.

Although having few registers and maintaining 8086 compatibility hurt its performance, Intel retained the 80x86 architecture throughout the 1990s. The 80586 P5 Pentium (1993) had more on-chip cache and was superscalar: it could start two instructions per cycle to use both integer ALUs, or its single FPU plus one ALU (*see* INSTRUCTION-LEVEL PARALLELISM). The MMX CPU (1995) added “Matrix Math eXtension” instructions to perform two to eight, 32- to 8-bit vector integer operations at once in its 64-bit FPU, speeding calculations for 3D graphic images.

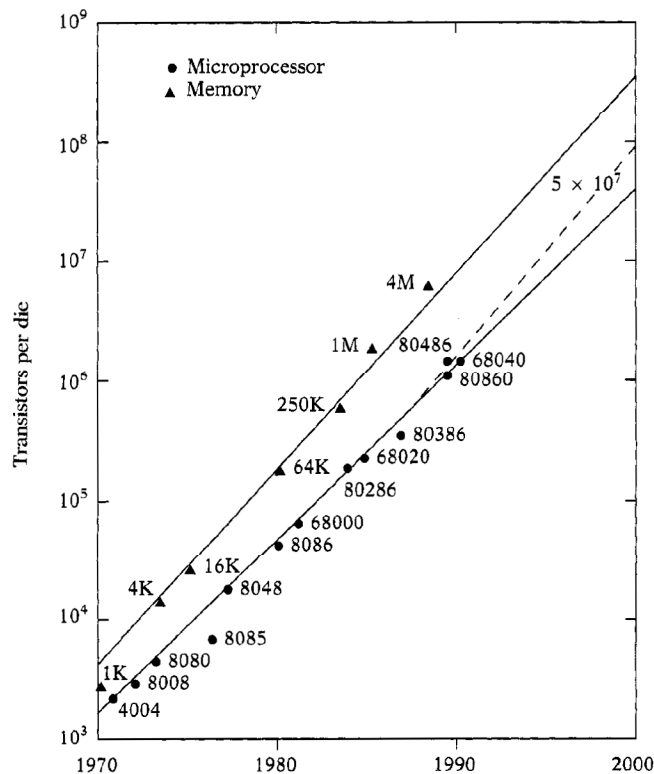


Figure 2. Almost annual doubling of transistors per chip for microprocessors and memories. The graph shows the densest memory chips and microprocessors from Intel (4044, 80xx, 80xxx) and Motorola (680xx). As microprocessors are built with more on-chip memory for fast access, the two lines will merge, as indicated by the dashed line (source: Intel Corporation).

The 80686 P6 Pentium Pro (1995) added level-2 (L2) cache on-chip and lessened pipeline delays after conditional branch instructions by speculative execution: guessing which instruction will follow a branch and starting to execute it immediately but not changing registers until the branch direction is known. The Pentium Pro did not execute its complex 80x86 family instructions directly; its decoding hardware translated them into simple instructions that pipelines could execute faster.

The Pentium II (1997) was a Pentium Pro with MMX instructions and an off-chip L2 cache, sold together on a heatsink module. Celeron (1998), a Pentium II with no L2, sold cheaply for under-\$1000 PCs. It was too slow; a small on-chip L2 was added late in 1998. The Pentium III (1999) added a large on-chip L2 cache. The first 64-bit Intel processor (Itanium) was due for release in 2000.

Fig. 2 shows the growth in transistors/chip from 1970 to 2000.

OTHER EARLY MICROPROCESSOR FAMILIES

Eight-bit microprocessors were important because they were the first to address enough memory and to be powerful enough for general-purpose computing. Zilog was founded by designers of the Intel 8080A. The Z80 (1976) was upwardly compatible with the 8080A, but faster. It had more instructions, more addressing modes, and an alternate register set for fast interrupt responses. Its support chips were less expensive. It was used for most personal computers before IBM adopted the 8088 for the PC. The Z80 is still successful in embedded applications. Its descendants—(Z8, Z180, Z280 controllers; 16-bit Z8000 (1979); 32-bit Z80000—are well respected.

The Motorola 6800 (1974) was very different from the 8080. It had two accumulators, a stack pointer, and an index register (*q.v.*), but no data registers; it relied on powerful addressing modes to process data efficiently in main memory. Programs for the 6800 were more compact than for the 8080, saving expensive memory and execution time. The 6802 added a small, fast, on-chip RAM. The 6809 was the last 8-bit architecture from Motorola. It added two registers for base-page relative addressing and introduced a very sophisticated instruction set to improve code density and execution speed. The 6809 is a good example of instruction set design based on measured instruction usage patterns.

The Motorola MC68000 (1980) was significantly different. It used a 16-bit data bus to access 32-bit values and had eight data registers, eight index registers, two security levels, an orthogonal instruction set (with operand addressing modes chosen independently of the instruction), and a nonsegmented 16 MB (2^{24})

address space. It was much more elegant and powerful than the 8086 of the prior year. It was adopted for many personal computers and embedded applications. The 68010 (1982) restarted instructions after page fault interrupts to support virtual memory. The full 32-bit 68020 (1984) had a 4 GB address range, an on-chip memory cache, and many powerful new instructions and addressing modes. It became the standard CPU for high-performance Unix workstations. The 68030 (1989) added an on-chip MMU to map virtual addresses. The remarkably powerful 68040 (1990) used large on-chip caches, integrated floating-point, and aggressive pipelining to run four times faster than the 68030, but was the last of its family, which was buried by 80x86, PowerPC, and SPARC CPUs for PCs and workstations.

Although the 6800 and 8086 families prospered, 4-, 8-, and 16-bit CPUs vanished from new personal computers by 1990. Many found niches as low-cost microcontrollers.

MICROCONTROLLER FAMILIES

For microprocessors, the 1980s were a time of architectural innovation, improving performance, and market consolidation. By 1980, hundreds of types of CPU chips had flooded the market. Eight-bit microprocessors dominated, but some 16-bit minicomputer architectures had been put onto chipsets used to build single-board computers. In 1973, National Semiconductor sold the IMP-16C, the first *single-board* 16-bit microcontroller. It was followed by the Digital Equipment (DEC) LSI-11 (1975) single-board version of the popular PDP-11 minicomputer, the Data General MicroNova board (1976), and the Texas Instruments (TI) TM990 card (1979).

Single-chip microcomputers include on-chip, all CPU, memory, and I/O subsystems, permitting very small, flexible, and inexpensive control circuits for consumer products. There are many controller varieties, targeting different functionality/performance/price points. In 1974, TI introduced the first single-chip 4-bit microcomputer, the TMS1000, which at \$2 was by far the most widely sold CPU in the world by 1980. The first Intel microcomputer chips were 8-bit: the 8048 (1977) with factory-masked ROM, 8748 with EPROM, and 8051 (1980). Microcomputer chips supplanted controller cards by 1980, but single-board computers reappeared in workstations in the late 1980s. Intel targeted the 16-bit 8096 (1984) and 32-bit 80960 (1987) microprocessors for fast embedded control, including automobile systems. The 80960 had on-chip floating-point, rapid context switching, instruction pipelines, superscalar execution, message passing, multiprocessing, and optional function units.

Microcontrollers based on existing microprocessors have inherently good software tools. The 68000 family includes the 68070 and 93C110 from Philips, plus the 68200 and 68300 from Motorola. Other designs focus on performance. The Harris RTX is a very fast, tiny 4,000-gate hardware implementation of Forth (*q.v.*), a popular real-time control language. Choices of on-chip support circuits and memory varieties determine suitability for many applications; many distinct controller chips coexist.

FAST REDUCED INSTRUCTION SET COMPUTER CHIPS

The late 1980s saw the rise of a new style of high performance CPUs, the single-chip RISC processors: the MIPS, Sparc, and PowerPC families. The key feature of these processors is efficient use of instruction pipelines made possible by easy-to-decode uniform code formats, large sets of registers for all arithmetic operands and results, and memory accesses limited to load and store operations. Other important features are that the entire CPU must fit on a single chip, since data exchanges are much faster on-chip than between chips; compilers must reorder instructions to keep pipelines busy; and alternative designs are evaluated by simulated execution of target programs.

The Control Data CDC 6600 supercomputer (1964) and IBM 801 chip prototype (1980) pioneered building fast computers using simplified instruction sets. The 6600 had many registers, a load-store architecture, simple three-address instructions executed in pipelines, multiple function units, and out-of-order instruction execution to avoid data conflicts. The 801 had 32 general-purpose registers, pipelined three-address single-cycle instructions, a delay slot to execute an instruction during each jump, and compiler instruction reordering. The 1981 Berkeley "Reduced Instruction Set Computer" (RISC) and 1982 Stanford "Microprocessor without Interlocked Pipe Stages" (MIPS) projects popularized simple instructions, pipelines, and code reordering. Limited to 100,000 transistors each in 1982, fast one-chip CPUs had circuit space for only the most useful instructions and hardware features. Early RISC chips had small, regular, hard-wired instruction sets, very different from the prevailing microcoded, complex instruction set computers (CISCs).

The RISC project succeeded. Hewlett-Packard began work in 1983 on a high-performance precision architecture (HPPA). Stanford engineers founded MIPS Computer Systems in 1984 and released the MIPS R2000 in 1986. In 1985, Sun Microsystems began designing its SPARC processor family based directly on the Berkeley RISC-II chip, and had CPUs by 1987. Many commercial RISC processors followed: Advanced Micro Devices (AMD) 29000 (1986), Motor-

ola 88000 (1988), Intel 80960 controller (1988), MIPS R3000 (1988), Intel 80860 graphics accelerator (1989), and the powerful "post-RISC" IBM Power-1 for the RS/6000 in 1989.

In the 1990s, RISC-style designs and shrinking IC device sizes greatly increased CPU performance: from 1.00 micron feature size (transistor length) in 1989 for the 25 MHz 32-bit CISC Intel 80486, to 0.18 microns in 1999 for the 480 MHz 64-bit RISC Sun/Texas Instruments UltraSPARC-3. IBM continued its 32-bit superscalar RS/6000 CPUs: Power-2 (1993) and Power-3 (1998), with eight function units allowing up to eight instruction completions per cycle. In 1992, IBM, Motorola, and Apple joined forces to redesign the Power-1 as a series of PowerPC CPUs to replace the 68040 and 80486 processors in Apple and IBM PCs: 601 (1993), 603 (1994), 604 (1995), and 750, marketed by Apple as G3, the third-generation PowerPC (1998).

In 1992, Digital Equipment (DEC) released its pioneering fast superscalar 64-bit RISC chip, the Alpha 21064 running at 150 MHz, then the 366 MHz 21164 (1995)—rated at 1,000 MIPS—and the 525 MHz 21264 (1998). In 1998, Compaq bought DEC in order to own Alpha. From the first, Alpha CPUs avoided the branch delay slots of MIPS and SPARC chips by speculative execution: guessing what instruction will follow a branch and starting to execute it immediately, but without changing register values until the branch direction is known. Delay slots work poorly for superscalar execution.

MIPS had a 64-bit R4000 chip in 1991, but its superscalar R10000 (1996) with on-chip FPUs was much faster. Sun Microsystems (design) and Texas Instruments (fabrication) released their 64-bit RISC SuperSPARC chip prematurely in 1992; it was slower even than Intel 80x86 CPUs. Their Russian-designed 64-bit UltraSPARC chip (1995) was much faster. It was superscalar (4 instructions issued per cycle) and had specialized instructions for moving I/O data blocks without disturbing the cache and for 3D graphics calculations.

FAST PROCESSORS AT THE END OF THE CENTURY

By 1999, the label RISC was a misnomer. Fast CPU chips held so many transistors—16 million in the Alpha 21264—that they could have the pipelines and general register sets characteristic of RISC chips as well as large CISC-like instruction sets and microcode to emulate older CPUs: PowerPC rapidly executes 68040 code (*see EMULATION*).

The PC and workstation processor market in 1999 was dominated by 32-bit 80x86 chips from Intel and three minor competitors: Advanced Micro Devices (AMD), Cyrix, and Integrated Device Technology (IDT). A few

high-performance 64-bit alternatives remained: 525 MHz Compaq Alpha 21264 (1998), 480 MHz UltraSPARC-3 (1999), and IBM-proprietary 200 MHz Power3 (1998) processors. The 32-bit 400 MHz Motorola PowerPC G4 (1999) was used in one PC in 20. Its fast AltiVec instructions render images 10 times faster than a G3. It is poised to supplant MIPS CPUs in game controllers.

Fast CPUs were a small-volume, but high-revenue part of the microprocessor market. In 1998, four billion microcontroller chips sold for \$13B (at an average price of \$3) plus one billion fast CPUs for \$19B—75% from Intel (at a \$20 average price).

The exponential IC density curve has slowed slightly to two years per doubling, but every decade seems to show another 10 to 15 years of growth left. Near-future CPUs will use smaller devices, more parallel execution, and larger caches on-chip to execute 20 times faster. Cryogenic cooling may further increase CPU clock rates to 25,000 MHz (25 GHz) by 2010.

Applications

Microprocessors are most visible as the central processors of general-purpose computing systems. Almost all computers, even highly parallel and expensive supercomputers, use commercial microprocessor CPUs because of their unmatched performance to cost ratio. Only the very highest performance computers have architectures too complex or use implementation technologies too sparse to fit the CPU on one chip. In 1999, only a few supercomputers used fast gallium arsenide (GaAs) circuits to maintain a lead. In 1999 most techniques used to improve supercomputers for the previous 30 years were being implemented in high-end silicon-based CPU chips.

The great majority of microcomputers are dedicated to controlling everything from consumer appliances to cars to space probes. In embedded applications, microprocessors serve as cheap replacements for otherwise very complicated control circuitry and often offer functionality not feasible any other way.

Microprocessors are playing an increasingly important role in transportation industries, from automobiles to airplanes to traffic regulation. The automotive industry has found a number of uses for microprocessors: controlling fuel and air delivery to the engine for optimal performance, fuel efficiency, and exhaust emissions; monitoring brakes to prevent skidding (anti-lock braking systems—ABS); controlling shock absorbers to give more comfort under varying road conditions and better handling by leaning into turns; tracking road surface markings to warn when a car strays from its lane; and processing traffic information to recommend shortcuts that avoid traffic jams. Micro-

processors also regulate traffic by counting local vehicles and adjusting signal light timings accordingly. In aircraft, there is an even greater reliance on lightweight microprocessors. They monitor and operate many flight systems, like inertial navigation, and report status to flight computers that display summaries on video screens. New planes increasingly “fly-by-wire” under direct computer control.

Personal electronics is a ready match for microprocessors. Toddlers of today have cuddly toys that talk and show images. Their older siblings can build mechanical pets that roam around and come when called. Many household electrical appliances such as washers and dryers have replaced complex electromechanical systems with simpler and more reliable electronic ones. Digital keypads on microwave ovens allow programmed cooking sequences; an unattended video cassette recorder (VCR) can tape several programs at different times on different channels; a digital chess game can provide a challenging opponent; digital wristwatches can calculate and play music; and telephones can store frequently dialed numbers. There are exercise stations that report calorie losses and monitor heartbeat to help maintain optimal exercise rates. Devices such as compact disc players and cellular phones, autofocus cameras, hand-held calculators, television game machines, and personal computers have been made possible by inexpensive microprocessors. Computer-controlled active jewelry is imminent and peek-a-boo clothing will come eventually.

Industry has embraced microprocessor technology. On the factory floor, microcomputers store shapes to guide automated manufacturing machinery, control component placement, guide robot arms that insert components, and keep track of parts inventory (*see* ROBOTICS). In buildings, microprocessors control air conditioners, turn off lights in empty rooms, and maintain security by logging all door openings and limiting access.

In offices, they handle small business accounting and inventory, generate payrolls, and run word processors. In home offices, they control personal computers, facsimile (fax) machines, and modems (*q.v.*) that increasingly let people work in pleasant locations instead of commuting to centralized offices. On the salesroom floor, microprocessors scan product bar codes, update inventory databases from point-of-sale terminals, and enable just-in-time inventory methods to keep overheads low. In the field, microprocessors control instruments and log data to be relayed to larger computers. In medicine, microprocessor-controlled scanners generate three-dimensional images of the body. In outer space, microcomputers control satellites, telescopes, and interplanetary probes.

Future applications will be even more wide ranging. The most interesting and revolutionary will be ones related to information management and sharing among a mobile population. Microcomputers connected to the Internet can distribute information, make large central databases easily accessible, and allow a wide exchange of ideas. Radio-linked, notebook-sized computer displays and wall-sized flat screens may supersede printed media. Palm-sized personal assistants with cell-phone modems already let travelers stay linked to the Internet (*q.v.*) to find local maps, exchange email, and receive interesting news from electronic distribution services. Pocket-sized cell-phones are ubiquitous. Infrared emitting badges allow office computer systems to find any worker rapidly. Virtual reality (*q.v.*) displays and video cameras will soon be mounted in everyday eyeglasses. High-definition television and three-dimensional virtual-reality extensions will be heavily dependent on fast microprocessors for local reconstruction of compactly encoded complex images.

Microcomputers and microprocessors are leading an epochal revolution in manufacturing, communications, information management, and personal services. As microprocessors grow in importance, they are shrinking in size and becoming nearly invisible parts of tools, appliances, and homes. They are vanishing into the walls of modern buildings, much as the indispensable electrical power network begun in 1900 has become an invisible part of the cultural fabric. Microprocessors are profoundly affecting society.

Bibliography

1977. Toong, H. M. D. "Microprocessors," *Scientific American* (Special Issue on Microelectronics), **237**, 3, 146–161.
 1989. Gelsinger, P. P., Gargini, P. A., Parker, G. H., and Yin, A. Y. C. "Microprocessors Circa 2000," *IEEE Spectrum*, **26**, 10 (October), 43–47.
 1989. Slater, M. *Microprocessor-Based Design: A Comprehensive Guide to Effective Hardware Design*. Upper Saddle River, NJ: Prentice Hall.
 1996. Hennessy, J. L., and Patterson, D. A. *Computer Architecture: A Quantitative Approach*, 2nd Ed. San Francisco: Morgan Kaufmann.

Websites

- Bayco, J. Great Microprocessors of the Past and Present. <http://www.cs.uregina.ca/~bayko/cpu.html>. Insightful vignettes on the distinguishing features of 40 great processors and their kin.
 Burd, T. CPU Info Center. <http://bwrc.eecs.berkeley.edu/CIC/>. Archive site with Web links for all major microprocessors and latest Silicon Valley Industry news.
 Gröner, J. Johannes' Computer Time Line. <http://home.t-online.de/home/Johannes.Groener/inhalt.htm>. Very complete list of computer names, dates, and makers worldwide, mainly in English.
 Karbo, M. B. Click and Learn. <http://karbosguide.com>. An illustrated tutorial on the history and organization of Intel PCs. Easy reading.

Polsson, K. Chronology of Events in the History of Microcomputers. <http://www.islandnet.com/~kpolsson/comphist.htm>. A detailed list of computer industry events with no explanations.

Slater, M. *et al.* *MicroDesign Resources, Home for Microprocessor Report*. <http://www.mdronline.com>. Insider bible for Silicon Valley computer industry; older articles are free.

Larry D. Wittie

MICROPROGRAMMING

For articles on related subjects see COMPUTER ARCHITECTURE; CYCLE TIME; EMULATION; FIRMWARE; HOST SYSTEM; INSTRUCTION SET; LOGICAL DESIGN; READ-ONLY MEMORY; REDUCED INSTRUCTION SET COMPUTER; and WILKES, MAURICE V.

Microprogramming is a technique used by designers to implement the control functions of a computer. Usually, it uses a read-only memory (ROM) to implement the instruction decoder. Microprogramming, as originally conceived by M. V. Wilkes in 1951, is a specific technique "to provide a systematic approach and an orderly approach to designing the control section of any computing system." In Wilkes' context, the term *control* is taken to mean the interpretation and execution of a machine instruction.

Microprogramming was widely used throughout the 1970s and 1980s as the preferred implementation of the instruction *decoder*. The IBM System/360/370 (*q.v.*) series and the DEC VAX systems are typical examples of microprogrammed control. The opcode of an instruction contained in the instruction register was used as an address (or part of an address) to a read-only control store. The contents of this address would define the first of a sequence of microinstructions or control words which specified the control values that implemented the required register-to-register data movements and ALU (*q.v.*) operations required for instruction execution. The regular implementation of the decoder through the use of storage provided a convenient way to change an instruction set by changing control store contents and also provided a debugging (*q.v.*) facility for designers. The availability of an extendible control memory also allowed for the execution of multiple separate instruction sets in a process called *emulation* of instruction sets. The IBM System/360/370 provided emulation of older instruction sets including the IBM 7090 and IBM 1401 (*q.v.*).

As the functions for microprogram control store increased, implementations changed so that the control store consisted of both a read-only portion and a read-write portion. This enabled certain microprogramming routines to overwrite a portion of the control store and facilitated functions such as microdiagnostics and fault location, as well as support for complex operating system functions and security checking.

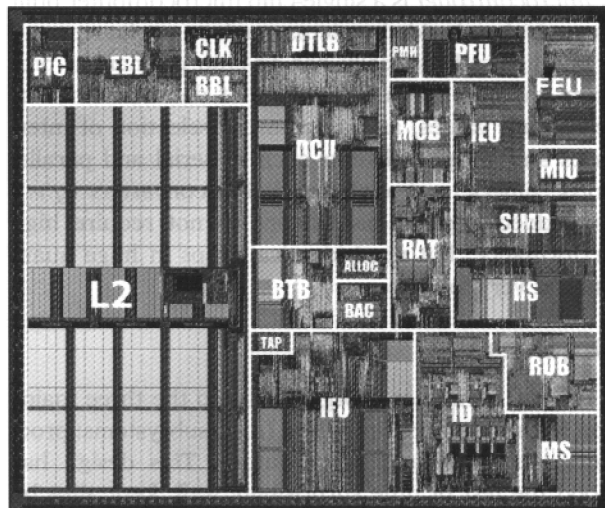


Figure 1. Photomicrograph of an Intel Pentium III processor introduced in 1999. It is a VLSI circuit with 28.1 million transistors. Among the major parts of the chip, marked with an overlay, are—left side: the cache memory (L2), and interface to the external bus (EBL); top center: data allocation units, including the data cache (DCU); bottom center: the instruction fetch unit and front-end of the instruction pipeline (IFU); top right: arithmetic execution units (IEU: integer execution), (FEU: floating-point execution), (SIMD: MMX technology instructions); bottom right: instruction decoder (ID) and associated logic for instruction scheduling. (Courtesy of Intel Corporation.)