Micromainframe Is Newest Computer on a Chip

What does one do with 100,000 or more transistors in a single integrated circuit? The next generation of microcomputers is one possible answer

There never has been a foolproof way to measure the power of a computer. In the end, cynics have suggested, all one can say is that large computers cost a lot of money and small computers cost much less. Nonetheless, one imperfect measure of computer power is the number of binary digits (bits) the machine works on simultaneously (word-width). A decade ago, when they were new, microcomputers had 4-bit word-widths but graduated to 8 bits within a year. Minicomputers mostly have had 16-bit words, although powerful 32-bit superminis were introduced in the mid-1970's. And large or mainframe machines had words of 32 to 64 or more bits.

The march of technology is bringing down this neat classification, however. Microcomputers that have 16-bit words and as much power as many minicomputers first appeared 5 years ago and have proliferated during the last 2 or 3 years. Then, last year, word began circulating that one company was planning on introducing a 32-bit microcomputer within a few months. Finally, at the International Solid-State Circuits Conference held this February in New York, four groups presented papers on this latest generation of computers on a chip. Actually, what the groups discussed were not full microcomputers but microprocessors—the central processor unit or CPU of a computer. Additional chips are needed for memory, input/output circuitry, and so on. The four were from the Intel Corporation, Aloha, Oregon; the Hewlett-Packard Company, Fort Collins, Colorado; Bell Laboratories, Murray Hill and Holmdel, New Jersey; and the National Semiconductor Corporation, Santa Clara, California. Intel and National will be selling their chips later this year. Hewlett-Packard and Bell Labs (by way of Western Electric) make microcircuits for their own use only and have not even said that the particular microprocessor chips described will ever be used in any product.

Of the four reports, those from Intel and Hewlett-Packard drew the most attention, but for somewhat different reasons. Intel is a leading example of a merchant vendor, a manufacturer of microelectronic circuits, also called integrated circuits, that other firms use to make computers and other electronic equipment. Hewlett-Packard is a producer of minicomputers and scientific instruments, but also designs and fabricates some of the integrated circuits it uses. As one observer from a rival semiconductor company said to *Science*, "Everyone knows Intel has the technology to make [advanced, ultraminiaturized circuits]. What they have to show is that they can design a computer. That is what they are doing here." Hewlett-Packard, on the other hand, "knows how to build computers. They are showing they have the advanced processing technology necessary to get into VLSI." VLSI—very large scale integration—is the generation of integrated circuitry the industry is moving into with 100,000 or more transistors on a single silicon chip.

There was additional applause for Intel's 32-bit microprocessor because the ambitious, 5-year project that the company undertook at a cost of $20 million (some say more) to develop the product seems to have succeeded in answering some questions that have been increasingly plaguing the microelectronics industry. As the pace of miniaturization continues at a nearly exponential rate, the resulting integrated circuits become more complex to design, to build, and to use. A good deal of the cheering emanated from Intel itself, which has been signaling its intentions for well over a year in the trade and business press. The company calls its creation a micromainframe because it has the performance of a large machine at the cost of a microcomputer. According to industry analyst Benjamin Rosen of Rosen Research, Inc., in New York, "all the puffery is probably valid. Intel is several years ahead of anybody else in the world."

The crux of the problem was graphically described by Intel president Andrew Grove to a meeting of New York security analysts early last year. In 1975, an 8-bit microcomputer system costing $100 required that the equipment manufacturer who used it in his products expend about $20,000 for the half-man-year of effort necessary to write a program. By 1980, a 16-bit microcomputer system with more than ten times the memory capacity still cost only $120. But, Grove estimated, $450,000 and the time of three persons each working at least 2 years would be absorbed in writing the much more complex program (software) needed. Extrapolating to a 32-bit machine, one can project another factor of 10 increase in software costs. In choosing a 16-bit rather than an 8-bit microcomputer, the user is motivated by the fact that the less powerful machine cannot perform well enough to carry out the intended application. Thus, the higher programming costs are unpleasant but necessary. Nonetheless, if they get too high in going to 32 bits, interest in the powerful but expensive chip will diminish, and the microelectronics revolution will have to find a new path.

Intel's answer to this dilemma is not unique. In a Texas Instruments presentation some time ago, the Dallas-based company's Egil Juliussen discussed the evolution of integrated circuits. The first solid-state electronics devices were discrete components, such as diodes and transistors. The first integrated circuits in the 1960's combined components to place such circuit functions as flip-flops and counters on a chip. In the 1970's, with the advent of large-scale integrated circuits, entire computer functions were put in one device (computer memories, microprocessors, and so on). The next step, said Juliussen, to take place in the 1980's is the implementation of software functions in integrated circuits; that is, in effect doing some of the program writing by building some of the software into the chip itself. Intel's new 32-bit micropro-

The higher programming costs are unpleasant but necessary.

0036-8075/81/0501-0527$00.50/0

cessor does this in at least two ways.

One of these is the incorporation of a high-level language. The instructions executed by the CPU of a computer are in the form of a sequence of binary digits, so-called machine language. Almost all microcomputers (and many larger machines) are programmed in a slightly more comprehensible (to humans) set of symbols known as assembly language. A special program (the assembler) translates the assembly language program into one the machine can understand. The programming cost comparisons made by Grove for small and large microcomputers assumed working in assembly language. A high-level language is one like Fortran or Cobol, in which instructions bear some resemblance to ordinary English. A compiler (another program) translates the high-level language program into one executable by the machine. In his talk to the security analysts, Grove foresaw a factor of 5 decrease in programming costs to the user if a microcomputer could be programmed in a high-level language.

Large computers that run programs of many different types have their own assembler and compiler software. Microcomputers not only run the same program over and over, but hundreds, thousands, or more chips executing the identical program may be fabricated. In this case, the assembler program is run once on a separate larger computer, and the microcomputers store the translated program in a type of memory having fixed contents called a read-only memory.

Intel's micromainframe skips the assembly language stage altogether; all programming is done in a high-level language called Ada (*Science*, 2 January, p. 31). As with assemblers, the compiler to translate Ada still runs on a separate, larger computer. This language, which originated in Europe and has become famous because the Department of Defense is adopting it as its standard working computer tongue, has several features that are reflected in the design of the microcomputer itself and that make programming the machine easier.

Among other things, Ada was designed to facilitate writing lengthy, complex computer software by means of an approach called structured programming. Stripped to its essentials, structured programming means two things. First, the program is divided into more or less independent modules that can be written by teams of programmers with relatively little communication between them. Second, the modules are constructed in such a way that only certain,

(Continued on page 530)

Magma Chambers in the Laboratory

Much of the earth's crust, especially the 70 percent that underlies the ocean, probably passed through magma chambers on its way up from the mantle. Petrologists have given much thought to a few processes, such as crystallization, that affect magma while it is held up in a chamber. But at a recent conference* on the generation of new ocean crust, specialists from a variety of fields reminded petrologists that a number of other physical processes may play important roles in determining the chemical composition of rocks.

One such physical process is convection. In an unusual collaborative effort, Stephen Sparks, a volcanologist at Cambridge University, teamed up with fluid dynamicists Herbert Huppert of Cambridge and Stewart Turner of the Australian National University in Canberra. Together, they created a magma chamber in the laboratory—sort of. To study what happens when a fresh batch of molten rock is injected into an aging chamber, they used concentrated solutions of potassium nitrate and sodium nitrate to represent the two magmas. They simulated convective mixing of the two batches of "magma" by adjusting the densities of the solutions to produce the density difference that they expect in chambers beneath a midocean ridge. In the past, petrologists had not taken much account of density differences and had assumed that new batches of magma mixed immediately.

At the start of their first experiment, the fresh hot "magma" injected at the bottom of their fish-tank–like "chamber" staunchly resisted mixing with the cooler fluid above it. It ponded at the bottom because its salt concentration had been adjusted to give the higher density expected of a magma frosh from the mantle. Because heat can diffuse faster than salts, both layers began to convect vigorously as heat crossed the interface between the layers, but the salts (the chemical components of magma) remained essentially unmixed. Instead, the diffusive cooling of the lower layer precipitated potassium nitrate crystals (the mineral olivine in a real magma chamber). Once the crystallization had decreased the density of the fresh "magma" sufficiently, the lower layer overturned and the entire "chamber" mixed. The significant aspect of that kind of mixing, the experimenters say, is that it did not happen until the composition of the magma had been changed by the precipitation of the crystals, which stayed behind on the bottom.

Some conference participants see these laboratory results as a possible way out of a quandary they are in. Experimental petrologists have been melting mantle-type rocks in the laboratory to determine where in the mantle the magma is formed that supplies the midocean ridges. One school holds that those magmas originate from 60 kilometers down or deeper. The quandary is that practically no rocks with the expected high magnesium contents are found in the ocean crust. The sort of fluid dynamics described above could act as a barrier to the ascent of dense, high magnesium magmas until they lose some of that magnesium as olivine and approach the composition of known crustal rocks.

Whether this or other laboratory simulations accurately represent physical processes in real magma chambers remains to be seen, but petrologists have been put on notice. Magma chambers will probably never be simple again.

*The Generation of the Oceanic Lithosphere, held at Airlie, Virginia, 6 to 10 April 1981, by the American Geophysical Union

More Deep-Sea Hot Springs in the Pacific

A French oceanographic expedition to the tropical section of the East Pacific Rise has confirmed that the bizarre "black smokers" on the northern Rise are not the only vents spewing hot, mineral-laden water from that midocean ridge. The presence of hydrothermal activity was about the only development that turned out as expected during this first exploration of a ridge that produces new ocean crust at a "superfast" rate.

Jean Francheteau of the Centre Océanologique de Bretagne in Brest reported that the cruise of the French research vessel *Jean Charcot* in the spring and summer of 1980 was excit-

0036-8075/81/0501-0528$00.50/0

(Continued from page 528)

well-defined types of information flow between them during execution of the program. For example, a program that tracks credit card accounts usually has to know in what kind of memory device the accounts are kept and the location in memory of each account. In an unstructured program, this information must be available every time an account is accessed. In a modular, structured program, only one module needs to keep this information. Every other module that manipulates an account in any way only has to know that it can be reached through the module that has this information. Besides making the program easier to write, modularity means that usually changes in the program can be confined to the module involved.

This characteristic of Ada is directly reflected in the design of the micromainframe by way of its object-oriented architecture. Architecture is computer jargon for design—how information flows through the machine as seen by the programmer. As one attendee at the solid-state circuits conference told *Science*, "To some extent, object-oriented architecture is as much a state of mind as a state of fact." Traditional microcomputers are filled with structures called registers, which are temporary repositories of

Tale of the Orphaned Genes

Not to be outdone by the particle physicists' penchant for anthropomorphic nomenclature, molecular biologists have now coined such a term of their own: orphon. The word has been affixed to a newly discovered type of gene that is plucked from the bosom of its multigene family and set down to rest in isolation elsewhere in the genome.

In reporting their finding in *Cell*,* Geoffrey Childs, Rob Maxson, Ronald Cohn, and Larry Kedes, of the Veterans Administration Medical Center, Palo Alto, suggest that the mechanisms that generate orphons might have interesting implications for gene evolution.

"We came across orphons quite by accident," Childs told *Science*. "We were looking for late histone genes in sea urchins—the ones that are switched on shortly after hatching—and instead we found orphons of the early genes."

There are five different histone genes (H1, H4, H2B, H3, and H2A), and in the sea urchin, early histone genes are arranged as many hundreds of copies of this quintuplet in tandem repeats. The late genes are of the same five types, but their nucleotide sequences are slightly different, there are far fewer of them, and they are not clustered as multiples of the neat five-gene groups. "The new histone genes we detected are scattered all over the genome, and they are clearly derived from the early genes," says Childs, who is now at the Albert Einstein College of Medicine, New York.

The discovery of orphons poses two questions. How did they arise? And what, if anything do they do?

Research in recent years has revealed genetic material to be in a surprising state of flux: chromosomes rearrange, and stretches of DNA hop about the genome with alarming alacrity. DNA mobility is often promoted by the intervention of transposons, relatively well-defined genetic units that are able to slip in and out of chromosomes, sometimes taking passenger DNA with them. Could peripatetic transposons be responsible for translocating copies of early histone genes from the multigene family to distant parts of the genome?

This is probably not the case, conclude Childs and his colleagues. The passage of transposons leaves a very distinct trail of nucleotide sequences in the chromosomes. Although there are some hints of such sequence arrangements associated with the orphons that have been studied in detail, most of the truly telltale signs are absent.

The secret of the histone genes' translocation may lie in their family arrangement of multiple tandem repeats. The pairing of chromosomes when cells divide frequently leads to the swapping of homologous pieces of DNA between the chromosomes. The phenomenon, known as crossing over, is particularly significant when gametes are formed, as this provides a source of heritable genetic variation. Areas of the genome that contain many repeated sequences are particularly vulnerable to unequal crossing over, because the match between DNA sequences can occur at many different points. The mismatch can produce a section of DNA that is looped out, excised, and discarded.

Instead of being discarded, looped-out histone genes may, suggest Childs and his colleagues, become integrated elsewhere in the genome, and thus form orphons. Support for this view of orphon origin comes from the observation that the other well-known tandemly repeated multigene family—the ribosomal genes—also has scattered isolated relatives.

The function of orphons may be potential rather than actual. The translocation of a gene from its original functional site removes it from its family influence, with two possible consequences. One, it escapes normal control and may eventually develop its own regulatory system. "This could lead to identical or very similar genes being expressed at different times in development," guesses Childs. "Perhaps this is how the separately regulated early and late histone genes arose."

A second possible consequence is that, once outside its original genetic responsibility, an orphon's nucleotide sequence is "free" to diverge, eventually encoding information for an entirely new product.

Childs and his colleagues go further and speculate that the many dispersed (as opposed to tandemly repeated) multigene families now known might have originated as multiple orphons. "A catastrophic deletion event could eliminate the parent cluster . . . leaving behind the dispersed orphons," they write. The now truly orphaned orphons would then assume a fully independent role.

The recently recognized promiscuity of genetic material has generated an atmosphere rich in speculation about many evolutionary mechanisms. The notions proffered by the Veterans Administration group are certain to be welcomed as further possibilities for contemplation. At the very least, however, the *Cell* report indicates that "there may no longer be a clear distinction between tandem and dispersed multigene families." Orphons now occupy that middle ground.—ROGER LEWIN

*Cell **23**, 651 (1981).

0036-8075/81/0501-0530$00.50/0

information such as data, instructions, and so on. The programmer must be keenly aware of what information is in which register at what time. In Intel's micromainframe, registers are still there, but the programmer does not need to know about them because of the object-oriented architecture.

Objects are entities stored in the computer memory. The data to be operated on are objects. The instructions to be executed are objects. And there are more abstract objects that have such information as which instructions are to be executed. One of these abstract objects is called a domain and corresponds to a module in a structured program. Like the module, the domain reserves to itself certain information that only it needs to know and it shares with other domains information that all can use. The programmer does not need to know where these objects are in memory or when any might be moved to a register. The microprocessor is designed to make these determinations by itself, once a call for a particular object is made.

An important capability given to the Intel microprocessor by its object-oriented architecture is called transparent multiprocessing. Multiprocessing refers to two or more CPU's operating in parallel to execute the same program. For multiprocessing to be done efficiently, some means must be found to divide jobs between the processors so that they all have something to do most of the time. This requires keeping track of all the data and instructions and feeding the right information to each processor as needed. To do this optimally, the programmer needs to know, among other things, how many processors are in the computer. Intel claims that the object-oriented architecture gives the programmer the ability to write software without knowing how many processors there are—the transparency—and that the microcomputer takes care of dividing up the jobs among whatever number of processors may be hooked together. At present, according to William Lattin, who was in charge of the 32-bit microcomputer project, up to several dozen may be connected, giving a performance spanning the range from a medium-powered minicomputer to a medium-powered mainframe, as measured by the speed of execution of a well-known test program.

The second major way in which Intel has incorporated software into its microprocessor in hardware form is through the operating system. Operating systems are programs that oversee the operation of a computer. A large computer with several remote, time-sharing terminals has to be able to decide which terminal to service first, whether to interrupt execution of one program and start on another with higher priority, and so on. Even in the execution of one program, the machine has to know when to call in an assembler or compiler, when to load the program and data into its memory, and similar things.

The micromainframe does not, of course, execute several programs in the sense of a general purpose computer center machine. But the one program a particular machine does run requires management of the objects that are the key to high-level language and multiprocessing. The essential part or kernel of the operating system that handles this job is implemented in integrated circuitry in a special form called microcode. Despite the similarity in name, microcoding is not unique to microcomputers; machines of all sizes may be microcoded or may not be. The basic operations out of which the instructions executed by a computer are constructed are defined by certain logic circuits and are therefore fixed. An alternative that is nearly as old as the computer itself is storing the basic operations in a read-only memory, so that they may be changed occasionally by replacing the memory or for other reasons. This approach is microcoding.

Intel's micromainframe is not all on one integrated circuit. The company introduced two chips, one containing some 110,000 transistors and the other 60,000, which together make up what the company calls a general data processor. Information flow through the processor is pipelined, a feature of large, high-speed machines that began to be popular in microcomputers when they grew to 16-bit machines. Pipelining means that processor operation is divided into several stages than can go on simultaneously, three stages in the case of the micromainframe. Thus, if there are three instructions in line to be executed, the second one can begin when the first instruction finishes the first stage, and the third one can begin when the first one finishes the second stage and the second the first, and so on. Speed is enhanced by the number of stages in the processor's pipeline.

To connect the general data processor to the rest of the world in the form of other microcomputers, mass memory devices, input/output terminals, or scientific instruments providing data, most of which does not as yet recognize object-oriented computer programming, Intel has devised a third chip, an interface processor, of about 65,000 transistors. The three chips together make up a microprocessor system which, when it becomes available later this year, will sell for about $1500, David Best, an Intel marketing manager, told a press conference held prior to the solid-state circuits conference. When sales volumes grow and experience in making the chips is accumulated, the price could drop to about $200 per set. For multiprocessing, additional sets of the two general data processor chips can be added.

One of the measures of the advancement of fabrication technology in microelectronics is the minimum dimension of features in the circuit pattern. Usually the width of the metal conductors connecting transistors is the standard for this determination. Intel's 32-bit microcomputer chips have minimum line-widths of about 3 micrometers, which is what the industry state of the art has been recently. The Hewlett-Packard microprocessor chip, however, was able to pack 450,000 transistors within its confines by reducing line-widths to 1.5 micrometers and the spacing between conductors to 1 micrometer. At one session, a spectator wondered out loud about the prospects of combining the computer architectual innovation of Intel's microprocessor with the technological virtuosity demonstrated by Hewlett-Packard's.

> " . . . object-oriented architecture is as much a state of mind as a state of fact."

Which brings up the last question, the traditional one asked whenever a new generation of computers is born: Who is going to use all that computer power? A recent news story attributes to rival Texas Instruments the notion that customers have barely had time to get used to 16-bit microcomputers and that a 32-bit machine is too powerful for most of their applications. The answer will probably be what it always has been. If the computer is inexpensive enough, applications no one dreamed of previously will appear. Intel, for one, has certainly bet a good deal of money that they will.

—ARTHUR L. ROBINSON