A coprocessor takes over floating point calculations and disposes of them in a fraction of the time needed by a μP relying on software algorithms.

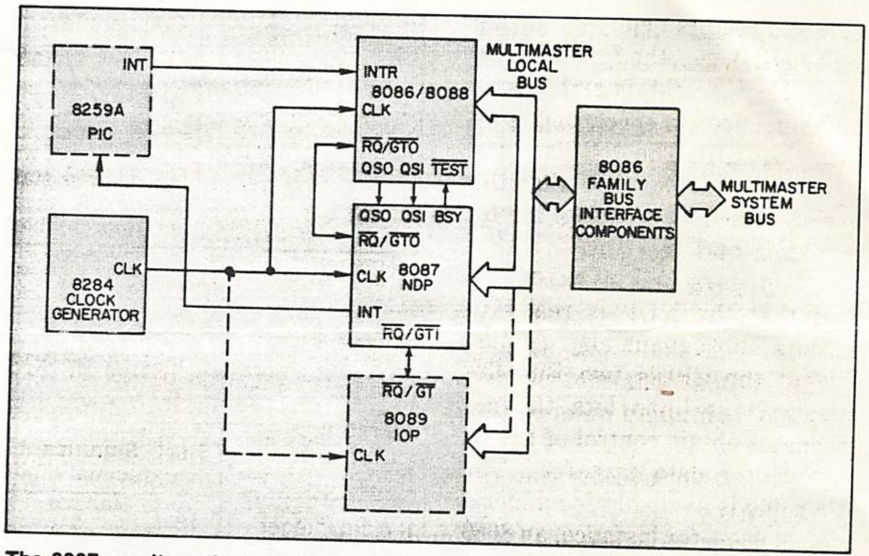
Coprocessor implements floating-point math

Increasingly, high-performance microcomputer systems rely on intelligent support chips to perform special functions. One of the latest of these chips is the Intel 8087 numeric data processor (NDP), which performs floating-point calculations one hundred times faster than a μP using software algorithms.

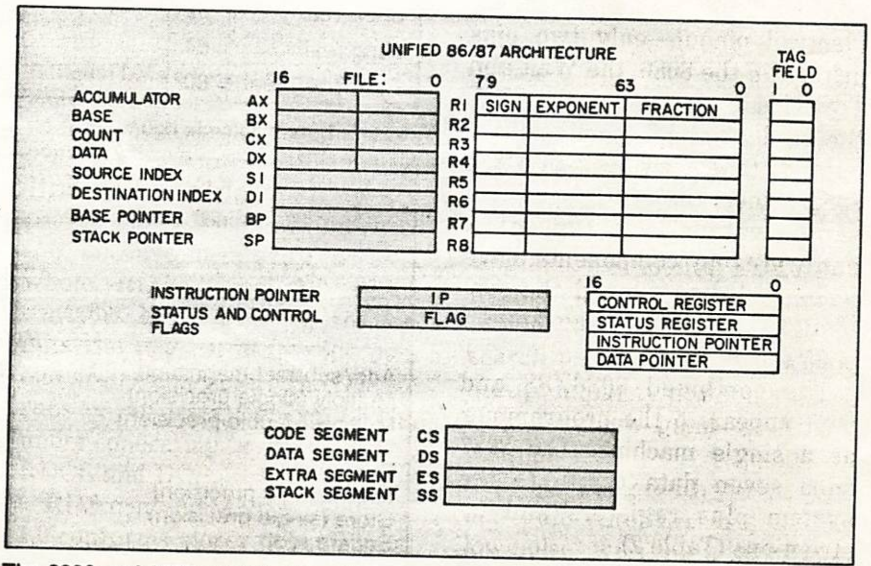
Previous support chips acted as peripherals to the CPU rather than as coexisting processors directly connected to the local bus. The 8087 serves as a coprocessor, monitoring the CPU's instruction stream and taking over when it recognizes certain specialized instructions.

First mentioned at the 1980 International Solid-State Circuits Conference, the device is fabricated with Intel's HMOS-1 process, and mates directly to an 8086 or 8088 microprocessor on a local bus.

The 8087 NDP implements the IEEE floating-point arithmetic format proposed by J.T. Coonan of the University of California at Berkeley. It handles single and double-precision formats, double-extended format, rounding control, precision control, infinity control and required instructions. The instruction set contains data transfer, arithmetic, logic, transcendental, constant and processor control operations (Table 1).



The 8087 monitors the instruction stream on the 8086/8088 local bus and is activated by an Escape instruction.



The 8086 and 8087, working together, appear as a single machine to the programmer. The 8087 adds seven data types to the combination.

An Escape instruction in a sequence of 8086/88 operations activates the NDP coprocessor and makes the 8086/88 perform a read to memory. The CPU ignores the data that come back. Meanwhile, a 6-bit operation code in the Escape command tells the coprocessor what operation to perform. The coprocessor accepts the data coming back and uses it either as data or as a pointer to the start of the data. The coprocessor has temporary control of the data and address buses for retrieving and storing data and results.

The NDP uses the 8086's queue status lines, to obtain and decode its instructions in synchronization with the CPU. The NDP's Busy signal informs the CPU that the coprocessor is executing an instruction, and the CPU Wait instruction tests this signal to ensure that the NDP is ready to execute subsequent instructions. One of the CPU's two Bus Request/Grant lines lets the coprocessor obtain control of the local bus for data transfers. The other line is available for general system use—for instance, an 8089 intelligent peripheral controller operating in its local mode.

Housed in a 40-pin DIP like the 8086/88, the 8087 has an almost identical pinout—only two pins differ. On the 8086, the Wait and Test pins are inputs; on the 8087, the equivalent pin positions are outputs that feed back into the 8086. Since the 8087 uses the same system lock and other hardware, no components have to be added to expand the system, as the block diagram on p. 35 shows.

The combined 8086/88 and 8087 appear to the programmer as a single machine. The 8087 adds seven data types, to the system plus registers and instructions (Table 2).

The NDP's register stack handles computations in eight 80-bit registers. The generous register space holds many constants and intermediate results during calculations, to reduce memory access and improve system throughput. The register stack can be accessed either as a stack (with instructions operating on the top one or two stack elements) or as a fixed register set, (with instructions operating on explicitly designated registers).

The NDP can be substituted in a preprogrammed system with little fuss. To develop programs with advanced math operations

Table 1. Principal 8087 instructions

Class	Instructions
Data transfer	Load and store (for all data types) exchange, free
Arithmetic	Add, subtract, multiply, divide, subtract reversed, divide reversed, square root, scale, increment, decrement, remainder, integer part, change sign, absolute value
Logical	Compare, examine, test, extract
Transcendental*	Tangent, arctangent, 2^x-1, y·$\log_2$x,y·($\log_2$ x+1)
Constants*	0, 1, π $\log_{10}$ 2, $\log_\theta$ 2, $\log_2$ 10, $\log_2$ θ
Processor control	Load control word, store control word, load status word, store status word, load environment, store environment, save, restore, set interrupt-enable, clear interrupt-enable, clear errors, initialize

*Combining these instructions in very simple routines provides all the common trigonometric, inverse trigonometric, hyperbolic, inverse hyperbolic, logarithmic and power functions.

Table 2. 8087 data types

Data type	Bits	Significant decimal digits	Approximate decimal range
Word integer	16	5	$-32,768 \leq X \leq +32,767$
Short integer	32	10	$-2.15 \times 10^9 \leq X \leq +2.15 \times 10^9$
Long integer	64	19	$-9.22 \times 10^{18} \leq X \leq +9.22 \times 10^{18}$
Packed decimal	80	18	$-99\ldots99 \leq X \leq +99\ldots99$ (18 digits)
Short real	32	7	$8.43 \times 10^{-37} \leq \lvert X \rvert \leq 3.37 \times 10^{38}$
Long real	64	16	$4.19 \times 10^{-307} \leq \lvert X \rvert \leq 1.67 \times 10^{308}$
Temporary real	80	19	$3.4 \times 10^{-4932} \leq \lvert X \rvert \leq 1.2 \times 10^{4932}$

Table 3. Execution time comparison

Instruction	Approximate execution time (μs)	
	8087 (5-MHz clock)	8086 emulation
Add/subtract magnitude	14/18	1600
Multiply (single precision)	18	1600
Multiply (double precision)	27	2100
Divide	39	3200
Compare	10	1300
Load (single precision)	9	1700
Store (single precision)	17	1200
Square root	36	19600
Tangent	110	13000
Exponentiation	130	17100

before the 8087 becomes available, programmers can use the company's floating-point arithmetic library (FPAL) for 8-bit processors. For the 8086, Intel provides the ASM-86 and PL/M-86, an assembler program and a psuedo high-level language. ASM-86 has directives for defining all 8087 data types and mnemonics for all instructions. The PL/M-86 aids program development, so the programmer does not need a detailed understanding of the CPU architecture.

In their final form, these programs employ Call instructions to access a subroutine that emulates the 8087 instruction. When the 8087 becomes available, the Call instruction is replaced by an Escape command and a Wait instruction, without any other change in the software. This eliminates all subroutines that emulate the 8087 operations and decreases execution time by up to 100 times (Table 3).

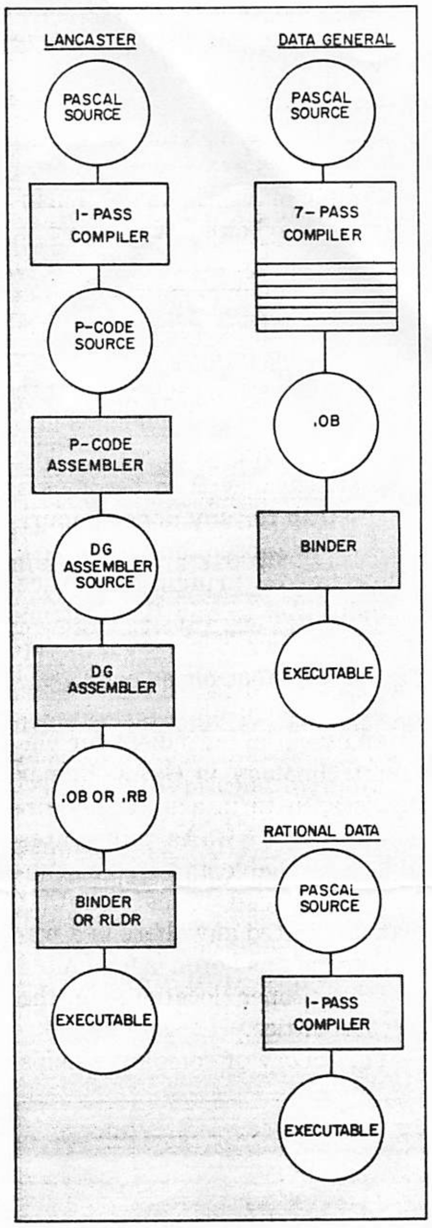
Dave Bursky

'Lazy input' refines interactive Pascal

One obstruction to the interactive use of Pascal—namely one-character look-ahead—is cleared away by a new implementation from Rational Data Systems (New York, NY). Input data stay in the input file buffer until the program actually asks for the buffer variable. An internal Boolean variable, Valid Window, is set false until the input line is complete. This lazy input does not violate the Pascal standard and is transparent to the user.

Since the RDS implementation needs no language modification, interactive Pascal programs remain as easy to read as any other Pascal code. Nor does the RDS solution depend on the operating system.

As a side effect, lazy input improves the efficiency of execution because lines are read as a whole. RDS Release 1.0, which contains the feature, also permits the user to define his own end-of-line character. The release runs on all Data General CPUs, under operating systems AOS, RDOS and DOS.



Rational Data's Pascal compiler is more direct than Data General's, but both are much simpler than the very popular Lancaster implementation.

RDS contends that its Release 1.0 conforms closer to the Pascal standard than Data General's MP/OS, which has only recently implemented nested procedures. On the other hand, MP/OS permits separate compilation, which RDS will implement later. Another major difference between the two implementations lies in the way that language is executed (see figure).

Data General's MP/OS creates threaded code in a seven-pass compiler, which has to pass through the Binder utility. Rational Data's single-pass compiler generates byte-oriented p-code which contains the needed library routines, so no binding is necessary. The latest release, 1.04, available April 28, processes about 900 lines/min.—similar to Data Generals's MP/OS.

Max Schindler

Processing discovery helps GaAs devices

Both the cause and the cure for performance and reliability problems in gallium-arsenide devices emerge from a new processing discovery by the Office of Naval Research (Arlington, VA). The probable cause: Dopant impurities tend to move about within the GaAs lattice. The cure: Dope the GaAs with impurities that cluster into aggregates too large to migrate in the lattice.

Max Yoder, Electronic and Solid-State Program Manager at ONR, presented this solution to the Semi-insulating III-V Material Conference in Nottingham, England, last month. He also announced that one manufacturer has already applied ONR research to build devices with the abrupt impurity gradients necessary for high gains at X-band frequencies.

Cooperating with ONR, Varian (Palo Alto, CA) has coupled low-noise with high associated gain in an X-band GaAs FET. Using a 0.2 × 50-μm gate, the Var-