FPS

UCRL-53745

The Livermore Fortran Kernels: A Computer Test of the Numerical Performance Range

Frank H. McMahon

December 1986

Lawrence Livermore National Laboratory

The Livermore Fortran Kernels: A Computer Test of the Numerical Performance Range

Abstract

A computer performance test that measures a realistic floating-point performance range for Fortran applications is described. A variety of computer performance analyses may be easily carried out using this small central processing unit (cpu) test that would be infeasible or too costly using complete applications as benchmarks, particularly in the developmental phase of an immature computer system. The problem of benchmarking numerical applications sufficiently, especially on new supercomputers, is analyzed to identify several useful roles for the Livermore Fortran Kernal (LFK) test. The 24 LFK contain enough samples of Fortran practice to expose many specific inefficiencies in the formulation of the Fortran source, in the quality of compiled cpu code, and in the capability of the instruction architecture. Examples show how the LFK may be used to study compiled Fortran code efficiency, to test the ability of compilers to vectorize Fortran, to simulate mature coding of Fortran on new computers, and to estimate the effective subrange of supercomputer performance for Fortran applications.

Cpu performance measurements of several Fortran benchmarks and numerical applications that correlate well with the cpu performance range measured by the LFK test are presented. The numerical performance metric Mflops, first introduced in 1970 in this cpu test to quantify the cpu performance range of numerical applications, is discussed. Analyses of the LFK performance results argue against reducing the cpu performance range of supercomputers to a single number. The 24 LFK measured rates show a realistic variance in Fortran cpu performance that is essential data for circumspect computer evaluations. Cpu performance data measured by the LFK test on a number of recent computer systems are tabulated for reference.

Correlation with FPS-264 Benchmarks

The Floating Point Systems FPS-264, a high-performance 64-bit processor for numerical applications, was first benchmarked using the LFK test in 1985. The harmonic mean performance of the 24 LFK on the FPS-264 is 0.74 of Cray-1 and 0.56 of Cray X-MP/1 performance (Appendix D). The harmonic mean usually correlates with and predicts the performance of partially vectorized "dusty deck" programs. The performance of the first LLNL applications converted to the FPS-264 have confirmed the performance level predicted by the LFK test measurements in Table 7. Notice that the range of speedups shown for the benchmarks is nearly identical to the subrange of LFK performance between the harmonic mean and the average.

A SISD/SIMD performance model is computed as a sensitivity test of the LFK test. (See the last table in Appendices B and C.) The model uses the harmonic mean of the lowest two LFK quartiles to represent the SISD scalar rate. The harmonic mean rate of the top LFK quartile is used to represent the SIMD vector rate. The net rate is given by the weighted harmonic mean, where the weight is the fraction of operations executed at vector rates.

Table 7. CPU performance ratios of LLNL applications— FPS-264 vs Cray.

Application	FPS-264/Cray-1	FPS-264/Cray X-MP
GEM2D	—	0.77
TOPAZ	—	0.53
DYNA2D	0.16	0.24
DYNA3D	0.13	0.21
12 benchmarks mean	—	0.53
24 LFK average	0.56	0.71
24 LFK harmonic mean	0.19	0.31

A deeper understanding of the performance of the FPS-264 relative to the Cray computers can

be inferred from the SISD/SIMD model in Fig. 7. This sensitivity analysis shows the complete harmonic variation of performance for all degrees of vectorization and for two vector lengths (VL = 468, 18). The cpu performance of vector computers

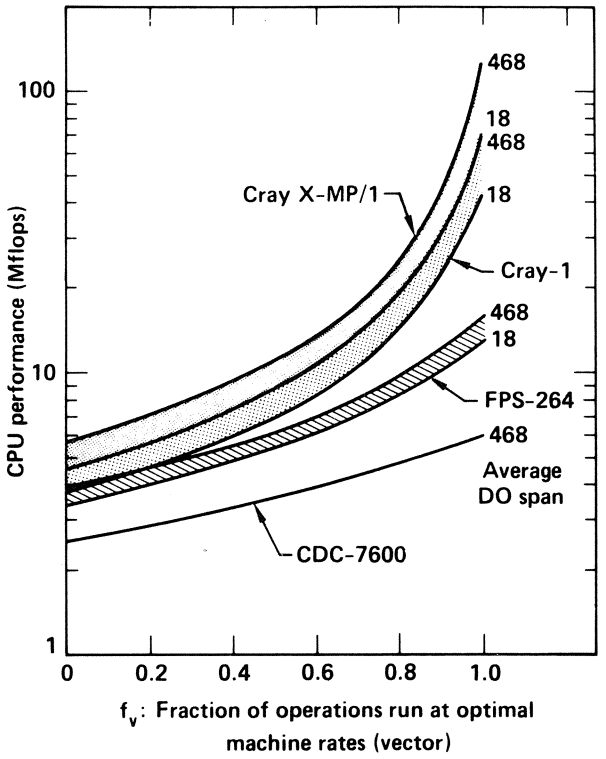


Figure 7. Performance ranges of the LFK on Cray X-MP/1 and FPS-264.

is seen to be a hyperbolic function of vectorization or optimization. The performance of most Fortran programs will be in the region bounded by the long vector and short vector performance curves.

The data shown in Fig. 7 imply:

1. The scalar performance of the FPS-264 is very close (90%) to Cray-1 scalar performance.
2. The vector performance of the FPS-264 is only four times scalar performance, which is typical of wide macro-instruction architectures. The Cray X-MP/1 vector performance is up to eight times faster.
3. The short vector performance of the FPS-264 is excellent and more efficient than Cray-1 short vector performance (Fig. 7).

To quantify the speedups, the performance ratios of the Cray to the FPS are plotted in Fig. 8 as a function of vectorization. The Cray computers have an impressive speed advantage of three to five times the FPS-264 for computations that are 95% vectorized. The Cray computer's advantage falls to less than a factor of two below 70% vectorization. When the vectorization is less than half of the computation, the Cray computers have a rather small performance advantage from 1.0 to 1.5 times the FPS-264. In this "dusty deck" domain (Fig. 8) the performance of the FPS-264 can fairly be described as Cray class.

The cpu performance ratios for several LLNL applications have been overlayed on the SISD/SIMD model in Fig. 8 to show the level of vectorization required for the performance measured. The overlayed data coincide with the peak of the performance range of the LFK SISD/SIMD model—in excellent agreement with the known, high level of vectorization in DYNA2D and DYNA3D. The correlation of the SISD/SIMD model using LFK measured performance and the performance of these complete application programs could not be better.

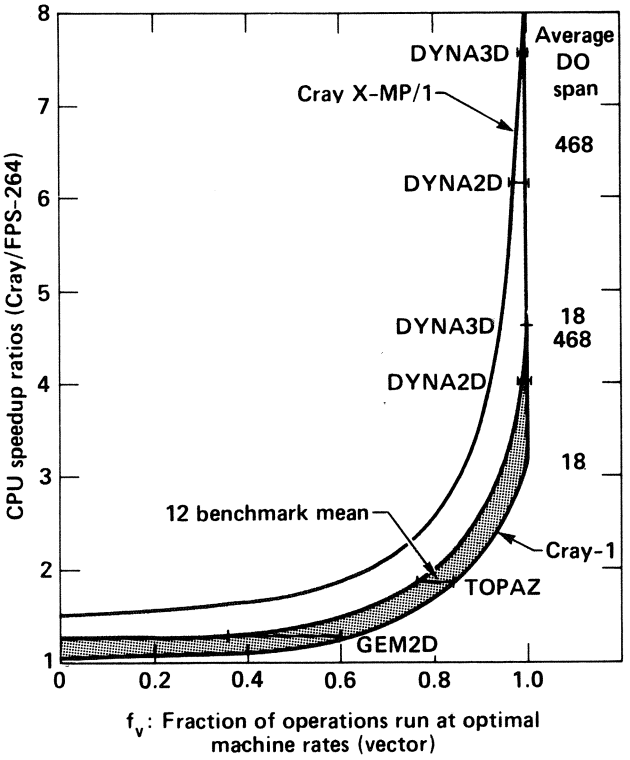


Figure 8. Performance ratios of the Cray X-MP/1 relative to the FPS-264.

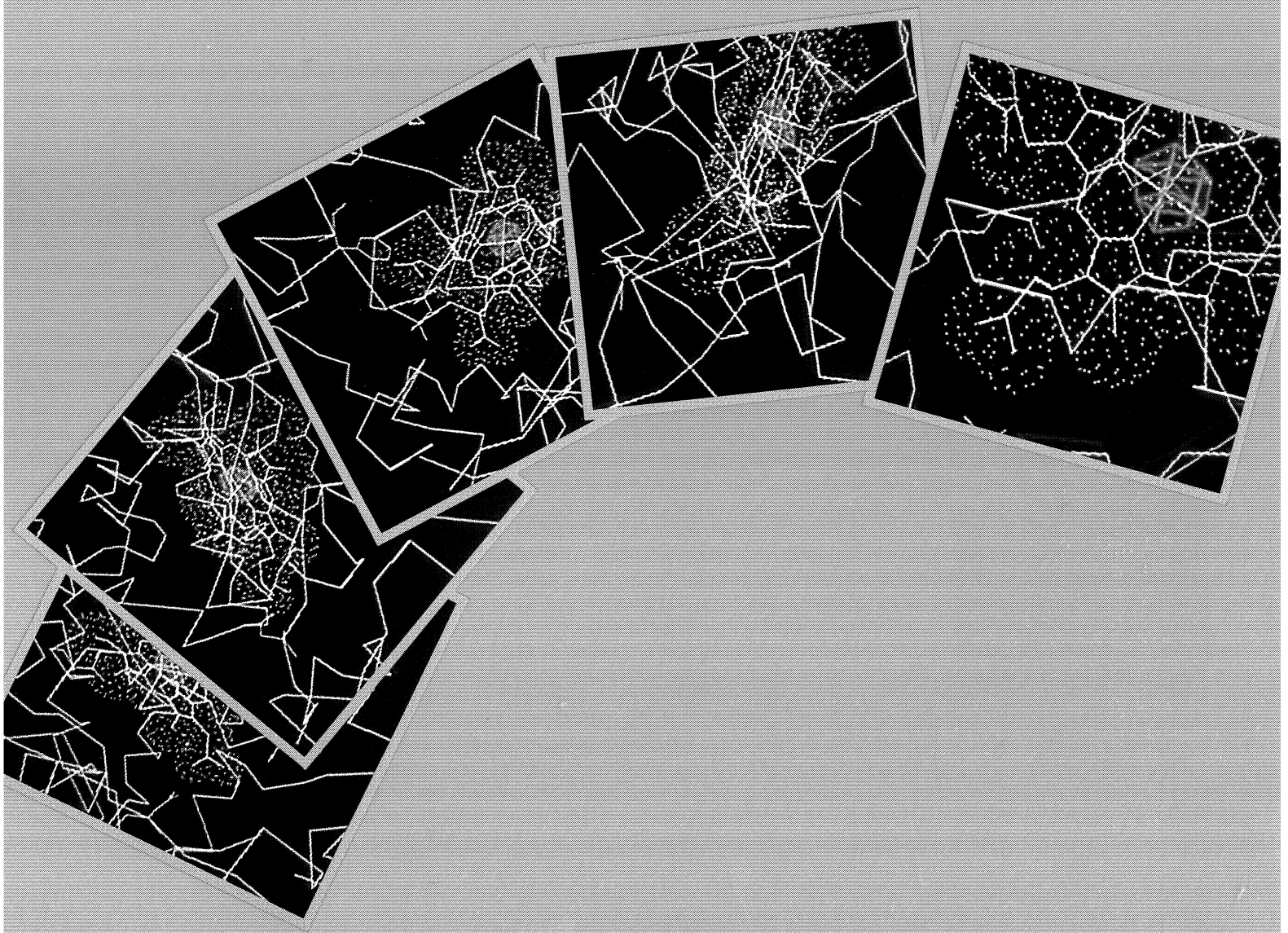
NOTE:
Current nomenclature for the FPS 264 Model Minisupercomputer is M64/60.

FPS Table of the CPU Performance
Ranges of the Livermore Fortran Kernels

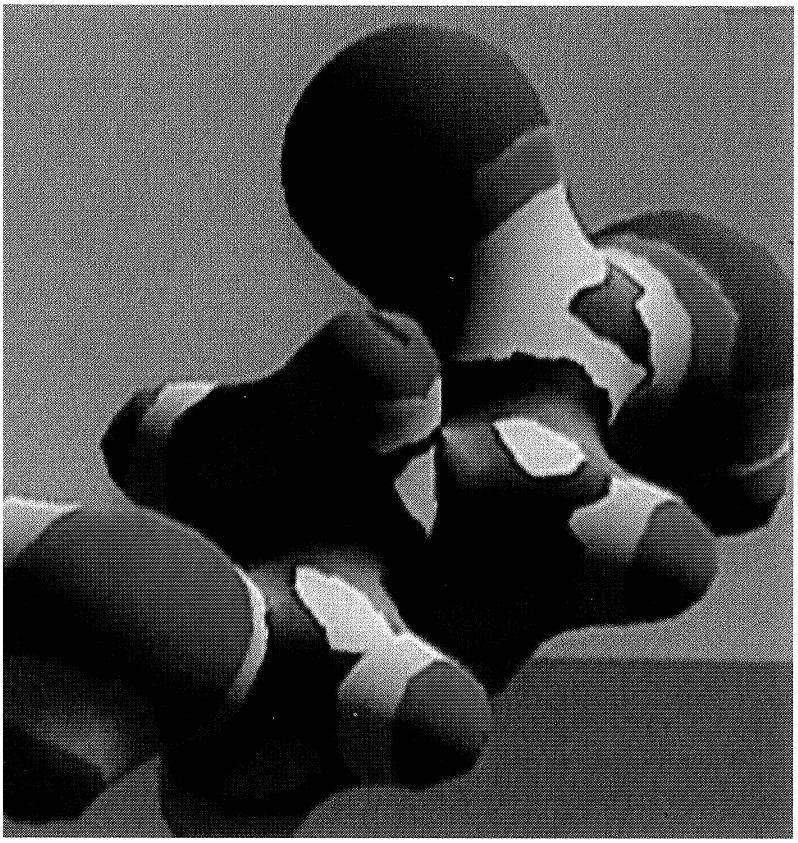
D.58 LFK Set#	58.1	58.2	58.3	58.4	58.5
Vendor	FPS	FPS	FPS	FPS	FPS
Model	164	264	264	264	264
ØSystem	SJE	SJE	SJE	SJE	SJE
Compiler		APFTN64	APFTN64	APFTN64	APFTN64
ØptLevel					
Samples	48	24	24	72	24
WordSize	64	64	64	64	64
DØ Span	236	18	89	166	468
Year	1983	1985	1985	1985	1985
Kernel					
1	4.41	15.51	18.04	18.91	18.91
2	2.08	4.22	7.37	7.37	7.37
3	3.52	10.26	12.00	12.60	12.60
4	3.21	4.13	7.89	11.40	11.40
5	1.77	5.65	6.16	6.32	6.32
6	2.66	3.93	8.03	6.74	6.74
7	6.05	18.51	20.96	21.64	21.64
8	3.93	12.37	13.96	13.41	13.41
9	5.51	17.18	20.76	20.76	20.76
10	1.41	4.66	4.97	4.97	4.97
11	1.76	5.20	6.00	6.30	6.30
12	1.53	3.37	6.03	6.30	6.30
13	1.01	3.17	3.63	3.71	3.71
14	1.63	5.42	5.74	5.55	5.55
15	0.83	2.75	2.67	2.61	2.61
16	0.54	1.91	1.87	1.83	1.83
17	1.46	5.45	4.93	5.22	5.22
18	2.60	8.39	9.53	9.53	9.53
19	1.99	6.13	6.96	6.95	6.95
20	1.08	3.73	3.72	3.82	3.82
21	1.98	4.69	6.79	7.17	7.17
22	0.91	2.75	2.79	2.79	2.79
23	2.33	7.70	8.88	8.88	8.88
24	0.36	1.23	1.24	1.24	1.24
PM Correlation =	1.00	0.98	0.99	1.00	1.00
Standard Dev. =	1.37	4.69	5.40	5.31	5.57
Maximum Rate =	6.05	18.51	20.96	21.64	21.64
Average Rate =	2.07	6.60	7.95	7.67	8.17
Geometric Mean =	1.68	5.29	6.34	6.09	6.47
Median Rate =	1.63	4.94	6.47	6.13	6.53
Harmonic Mean =	1.34	4.28	4.90	4.74	4.95
Minimum Rate =	0.35	1.23	1.24	1.23	1.24
Maxima Ratio=	1.00	3.06	3.46	3.58	3.58
Average Ratio=	1.00	3.19	3.84	3.70	3.95
Geometric Ratio=	1.00	3.15	3.77	3.62	3.85
Harmonic Ratio=	1.00	3.19	3.65	3.53	3.69
Minima Ratio=	1.00	3.49	3.52	3.49	3.52

FLOATING POINT
®SYSTEMS
Floating Point Systems, Inc.
Box 23489
Portland, Oregon 97223
Telex 4742018 FLOATPOIN BEAV
1-800-635-0938
Sales offices worldwide.

FPS AND THE COMING OF AGE OF COMPUTATIONAL CHEMISTRY.



FPS FLOATING POINT SYSTEMS



FPS DELIVERS SUPERCOMPUTING POWER TO MORE THAN 60 ACADEMIC AND INDUSTRIAL SITES.

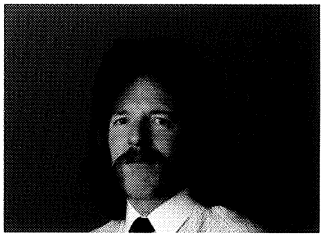
Mathematical alternatives to experimental chemistry have been proposed since the rise of quantum mechanics some 60 years ago.

Only within the last decade, however, has compute power become fast enough, and practical enough, to help chemists solve molecular problems of significant complexity.

And only now are techniques of computational chemistry moving from academic research to industrial application, including applied research and testing in medicinal chemistry, molecular pharmacology, chemical kinetics, agrochemistry, semiconductor technology and other related fields.

Floating Point Systems is proud of the pivotal role it has played in the development of computational chemistry. Chemists at the university and industrial level are using FPS minisupercomputers—and soon, the new FPS T-Series supercomputer—as the systems of choice for their pioneering and production work in this field.

FPS has continually set price/performance standards for compute-intensive machines. Now, for example, FPS minisupercomputers are affordable at the department level. They combine large memories with balanced scalar, vector and



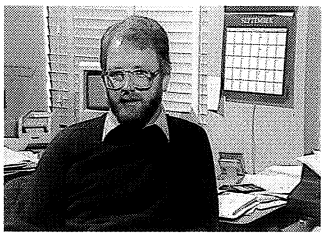
"In the 1986 we demonstrated the ab initio modeling of peptides (1000 basis functions, 200 atoms) on an FPS-164 minisupercomputer. The next generation of FPS machines, encompassing both parallel and vector technology, will provide the necessary hardware for making such calculations routine."

Martyn Guest, Ph.D.
Daresbury Laboratory of Science and Engineering Research Council

Quantum chemist Dr. Martyn Guest is using an FPS M64/145, FPS T Series and other supercomputers (Cray X-MP/48 and Cyber 205E) to develop ab initio quantum chemistry codes.

Dr. Guest's major milestone to date has been to develop ab initio capabilities to perform SCF calculations of up to 1000 basic functions on systems with no symmetry, specifically for the immuno-suppressive cyclic undecapeptide, cyclo-sporin A. The calculation was performed on a methyl-derivative, with the molecular geometry derived from the corresponding iodo compound.

Dr. Guest ran the calculation, believed to be the largest quantum chemistry calculation ever completed to date, on the FPS M64/145.



"It's really only been with the advent of such machines as the FPS, Cray and Cyber 205 that one starts to approach the kind of computational power necessary to tackle these problems."

Dr. Thom H. Dunning, Jr., Ph.D.
Argonne National Laboratories

Dr. Dunning and theoretical chemists at Argonne are investigating elementary chemical reactions involved in the oxidation of hydrogen and simple hydrocarbon fuels. Their goal is to obtain a quantitative description of the rates of these reactions, as well as build a qualitative understanding of these fundamental chemical processes, for application in engine design and many other fields.

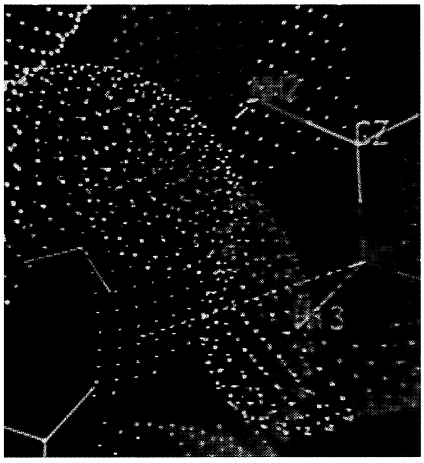
In determining the energetics of these chemical reactions, ab initio Multi-Configuration Self-Consistent Field (MCSCF) and Configuration Interaction (CI) methods with Gaussian basic functions are used to compute potential energy surfaces. Approximately 100,000 lines of Fortran comprise the main body of code in this ambitious and only recently feasible undertaking.

"In the last five to six years, advances in quantum chemistry, in the speed of computers and the development of code has allowed us to apply theories and models to actual applications."

Dr. Joseph M. Norbeck, Ph.D.
Ford Motor Company

Dr. Norbeck, Manager of Computer and Instrumentation Research at Ford, uses computational methods to study how emissions from vehicles interact with the atmosphere and create pollution.

There has always been an argument where theory and experiment in basic gas phase



matrix processing. Advanced FPS matrix algebra accelerator architectures feature up to 15 MWords of memory and can be custom-tuned for the needs of computational chemistry. Third-generation FPS disk devices provide more than 20 GBytes of high-speed storage.

On-staff chemists have helped FPS design hardware and software that free scientists from the more labor-intensive aspects of programming, allowing them to concentrate on the applications at hand. For example, FPS offers an extensive library of optimized matrix, simulation and mathematical subroutines, plus optimizing Fortran compiler, linkers, loaders and debuggers.

In addition, the leading third-party packages—such as DISCOVER,™ AMBER and GAUSSIAN 86—run on FPS machines. FPS aims to support chemistry with the most complete and current library of third-party codes.

For chemical science, this powerful coupling of fundamental theories with supercomputing technology makes it practical to cross barriers considered insurmountable just a few years ago. Starting with a theoretical model of the structure of individual atoms or molecules, and of the nature of the forces between them, chemists can use the laws of basic electrostatics, classical physics, quantum and statistical mechanics to describe the outcome of an experiment without actually performing it.

This approach is invaluable in solving problems that simply cannot be performed experimentally—those involving highly corrosive substances, for example, or particles that exist only briefly or at extremely high temperatures.

Moreover, by solving within hours iterations of ab initio, semi-empirical and molecular mechanical calculations that typically take general-purpose computers days and weeks to complete, FPS supercomputers can save tremendous time and cost at various stages of research.

Complex 3D and animated computer graphic simulations take a quantum leap forward from mechanical models or computer printout in bringing abstractions to life. FPS workstations within its M64 Series of minisupercomputers offer some of the most advanced graphic display technology in the industry.

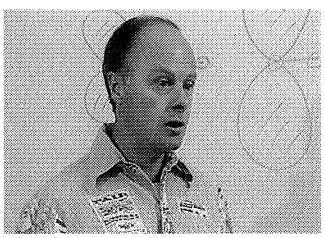
Computational chemists agree that their methods have transcended the experimental stage into commercial viability. The accompanying profiles of five FPS customer applications only begins to suggest the range of possibilities and wealth of experience that exists for advancing the science of chemical modeling and for accelerating results.

For further information specific to your application, and for an introduction to the family of FPS supercomputers, minisupercomputers and matrix algebra accelerators, contact Floating Point Systems or your local FPS sales office.

kinetics don't match, Norbeck says. The new computational methods are helping resolve these differences.

Norbeck's lab has developed a very elaborate Fourier transform infrared spectrometer "for continuously measuring all the different hydrocarbons coming out of an automobile exhaust pipe." By converting the analysis to an FPS machine, taking advantage of FPS' fast Fourier transform subroutines, the lab has cut its tests from 7-12 hours to about 9 minutes each.

"Now that," he says, "is a spectacular turnaround!"



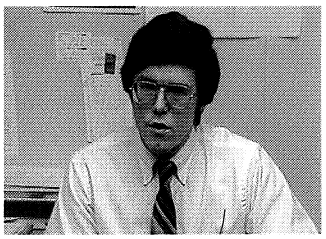
"These things were dreams five and certainly ten years ago that I don't think people felt in a realistic sense the private sector would be putting money into—but it is!"

Dr. John P Simons, Ph.D.
University of Utah

Dr. Simons and his group are using computational chemistry to refine the Born-Oppenheimer approximation of electron movement in molecules to make it more consistent with observed experiments.

In exploring the heretofore neglected relationship of vibrational motion to electronic motion, Simons requires the "brute force crunching" of calculations on FPS and Cray supercomputers to learn more about the strength of the coupling force and rate at which electrons should be ejected from particular molecules.

"The whole point here," says Simons, "is that theory people should push their models further and further as the experiments become more and more precise.



"Monsanto has a major commitment to exploring CAMD technology, and so we have purchased the Floating Point Systems computer hardware... we are gearing up to train about 20 more researchers in the use of the computational and graphics systems."

Dr. Henry E. Dayringer, Ph.D.
Monsanto

Dr. Dayringer is applying FPS machines in problems involving how particular proteins fit into receptors.

Computational chemistry, Dayringer explains, "helps by allowing us to explore the flexibility of a molecule and receptor, or just to compare a whole series of molecules to see how they may differ."

By saving steps in finding active compounds, the process can help trim time off the typical 10-15 year road from conceptualization of a pharmaceutical product to FDA approval—and optimize the effectiveness of the final compounds.

Ultimately, Dayringer says, computational chemistry will be accepted as "just another tool," like cloning or gene splicing, at the command of the molecular biologist—a tool particularly adept at quickly quantitating the instincts and hunches of the researcher.

THE COMPUTATIONAL STORY: NOW ON VIDEOTAPE.

These and other leading computational chemists are featured on an 18-minute videotape exploring the emergence of this new science in applications ranging from ab initio quantum chemistry to macromolecular methods. Several state-of-the-art 3D, static and animated computer graphics sequences are also featured.

The tape is available in 1/2″ VHS format for a non-profit fee of $15. To acquire a copy, write on your letterhead to Floating Point Systems, Inc., Computational Chemistry Video, P.O. Box 23489, Portland, OR 97223.

FLOATING POINT SYSTEMS

Contact
Floating Point Systems
Box 23489
Portland, OR 97223.
Telex 360470
FLOATPOIN BEAV.
Sales offices worldwide.
Call 1-800-635-0938.

Graphics courtesy of Dr. Robert Langridge, University of California, San Francisco Computergraphics Lab, and Dr. George D. Purvis, University of Florida Quantum Theory Project.

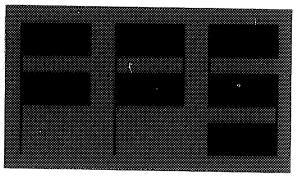
FPS MC SEC 60028, 4/87, 5M BR

FLOATING POINT
SYSTEMS, INC.

The age of array processing is here.

Never before has a computer actually delivered so much power, speed, precision, reliability, and programming ease . . . at such low cost.

The exploding growth of technology has created an insatiable demand for extremely high-speed, low-cost computing. FPS has responded to this need by developing a specialized scientific computer. Called an array processor because of its ability to deliver high throughput on large arrays or vectors of data, it has literally created a new era in signal processing and scientific computation. An FPS Array Processor interfaced to a minicomputer or large computer has created low-cost systems capable of handling the most demanding computational tasks previously achieved by only a handful of expensive large computers.



a new power in scientific computing.

- FPS Array Processors answer technology's continual demands for more computing power.
- FPS Array Processors allow systems to be upgraded at a small incremental cost.
- FPS Array Processors give new capability and extended life to large computers.
- FPS Array Processor Systems offer the best investment in original equipment.

an established company in scientific computing.

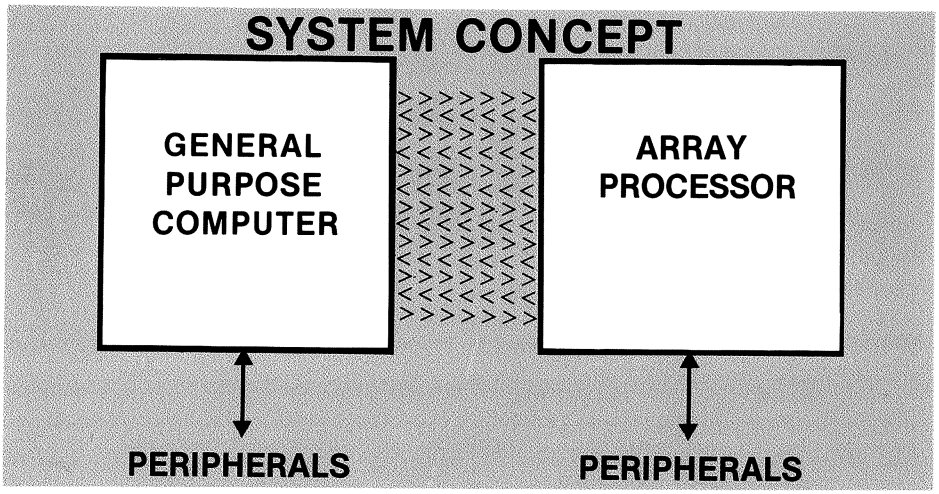
- FPS customers get unequaled software and support.
- FPS serves the needs of scientific computing technology.

Every day **numerically intensive array processing techniques** are applied to new areas of research and engineering. More precise mathematical models are created as data is acquired in greater quantities and expressed for machine processing in arrays. This vector approach to applications combined with a unique architecture and instruction set allows FPS Array Processors to offer computational speed and precision heretofore available only at great expense.

The benefits offered by Floating Point Systems and FPS Array Processors have been demonstrated in numerous areas from research to engineering, some of which include: the analysis of data in **seismic exploration** . . . the calculation, reconstruction, and enhancement of **images** (X-ray, satellite, seismic, and the composition of **images** in radio astronomy) . . . the conversion of **speech** signals to compressed digital data and their subsequent resynthesis . . . **vibration** and **structural** analysis . . . the processing of the hundreds of parameters that must be monitored in **nuclear reactors** . . . the analysis of the myriad data affecting **economic models** . . . the **simulation** of multi-variant airframes and environments, and more.

Introducing array processors by Floating Point Systems, Inc. Investigate their specifications and you will discover the computational power of large scientific computers at a fraction of the cost. Join our list of satisfied customers . . . It's growing daily.

Computational power where it's needed.



An array processor is a high-speed arithmetic processor. It takes data and instruction signals from a front-end computer/controller, performs the mathematical computations indicated by the instructions, and provides for the return of the "processed" data to the appropriate device. Combined with minicomputers, it has increased the computation speed of systems **by factors from one hundred to two hundred.** Combined with large computers, it provides computational ability previously available only at tremendous cost.

As original equipment.

The capital investment for computer power has been reduced by more than a factor of ten, so that an economical minicomputer and the Floating Point Systems Array Processor combination is a far better investment than a large computer in many original-equipment applications.

The array processor fulfills an urgent need for low-cost, high-speed computational power demanded by today's growing technology.

The age of array processing is here!

As an upgrade.

The investment in original equipment need not be increased significantly to upgrade a system. Adding an FPS Array Processor with extensive software library, documentation, and support to a minicomputer system normally calls for an investment of **less than $50K.** Adding an FPS Array Processor with equivalent library, documentation, and support to the typical large computer system can be accomplished with an investment of less than $100K. Thus, computing power previously found only in large expensive systems is available to a wide range of individuals.

Why vector data processing.

Most algorithms used to implement scientific models and their associated data sets are naturally structured in an array or matrix (vector) form. While the conventional computers of today require restructuring of the models, the architecture and instruction set of the FPS Array Processors have been designed to take full advantage of this natural structure. Furthermore, the use of one or more array processors for a scientific task lends itself to straight-forward division and distribution of the task for maximum system throughout.

Distributing the task.

Since a conventional general purpose computer must do several jobs well, the "task" is typically organized to fit the structure of the computer at the cost of programming complexity and time. This distribution of tasks is particularly unfavorable in scientific data processing, since the degree of sophistication of software and cost of manpower for scientific programming has caused the price for software to steadily rise, although the evolution of microelectronics has allowed the price of hardware to steadily decline. A computer system employing an FPS Array Processor takes cost-effective advantage of these concepts by allowing the array processor to efficiently take over the heavy computation, while the general purpose computer handles system peripherals, user interaction, and overall control. This special function concept of distributing the task to specialized components is further employed inside the array processor, where floating-point computation is handled by a specialized arithmetic section, and tasks to support the computations (internal data transfer, memory access, program execution, and processor control) are handled by special hardware structures. Thus, **FPS utilizes the most cost-effective hardware-software combinations,** to reduce the overall dollar-for-throughput cost to the user.

The move to floating point.

Previous to the FPS Array Processor, the few array processors in existence were largely integer-arithmetic devices, since the slower floating-point arithmetic of several years ago was undesirable when working on numerically intensive problems and/or large arrays of data. However, integer arithmetic made programming awkward, due to the limited dynamic range of the word length. Also, array scaling and "block" floating-point techniques allowed human and other errors to creep into the results, and were costly and time consuming. Then, as processing became more sophisticated, even 16-bit integer data words were not sufficiently precise for preserving the accuracy of simple 8-bit analog-to-digital converted input data.

With the advent of faster digital circuit elements, floating-point processing became as fast or **faster than previous integer processing,** allowing users the advantages of easier programming, wider dynamic range and greater precision.

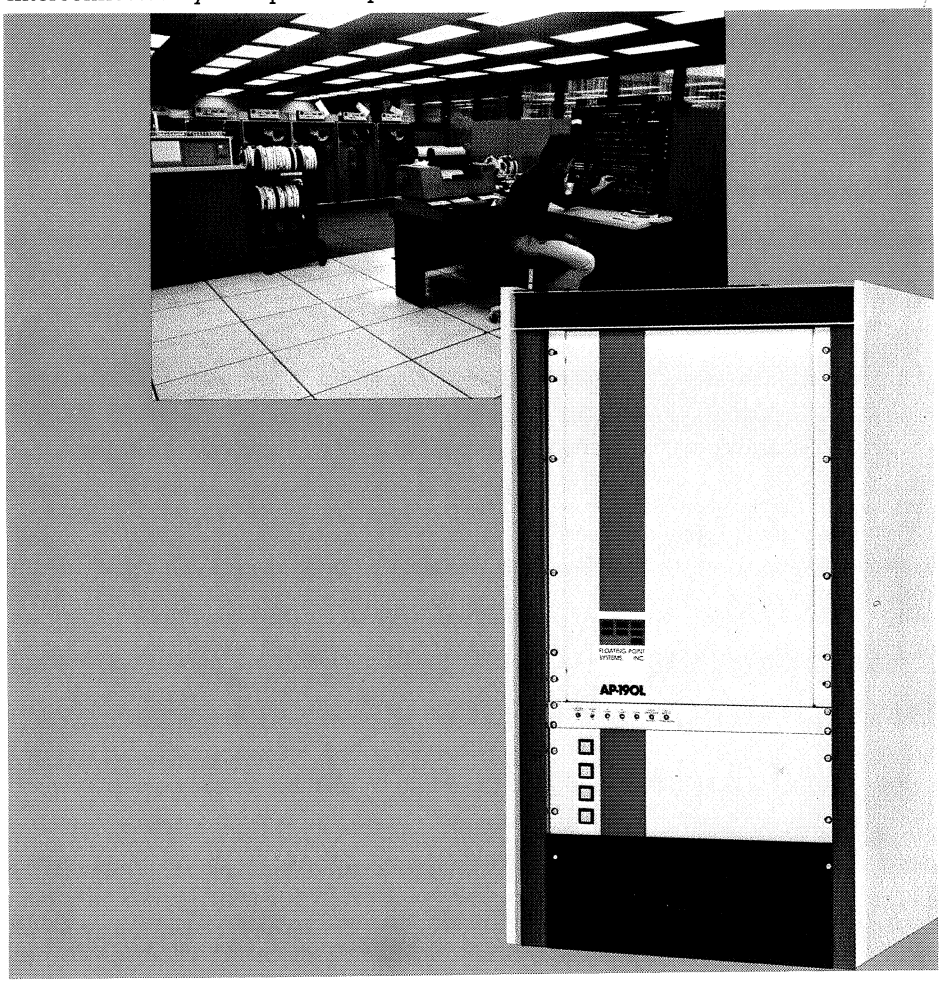
In recognition of this trend, Floating Point Systems, Inc., was formed in 1970 to specialize in floating-point array processor computers.

Preserving precision.

As a computational resource in a computer system, the array processor is called upon to compute on different source data formats. Consequently, Floating Point Systems became expert in format conversion "on the fly" (as data is being passed to or from the array processor), so processing time would not be used for format conversion. Why convert formats? Simple. Not all formats are mathematically clean. For example, it is unwise to use a 32-bit hexadecimal format for serious number crunching because a hexadecimal normalization can cause as many as three leading zeroes between the binary point of the mantissa and the first significant bit — this reduces precision.

The anatomy of an array processor.

Array processors from Floating Point Systems, Inc., are programmable, parallel, synchronous pipelined processors consisting of a number of fast registers, program source memory, data memory, table (or constant) memory, floating-point adder, floating-point multiplier, and an integer address calculator/indexer, all interconnected by multiple data paths.



Combined with a major mainframe, the FPS Array Processor brings the user computational ability previously available only at tremendous cost.

The FPS solution is to use a high-precision, 38-bit floating-point format. The format has a **28-bit mantissa,** plus 3 guard bits, which provide enough bits to not only allow for host hexadecimal formats, but also to carry enough information to permit extensive calculation without significant truncation error when the results are finally converted back to the usual 32-bit formats. The multiplier and adder allow full-width results, and perform post-normalization and convergent rounding at the end of each arithmetic operation in the array processor.

FPS also uses a 10-bit binary exponent, which has more dynamic range than the standard 7-bit hexadecimal or 8-bit binary exponent.

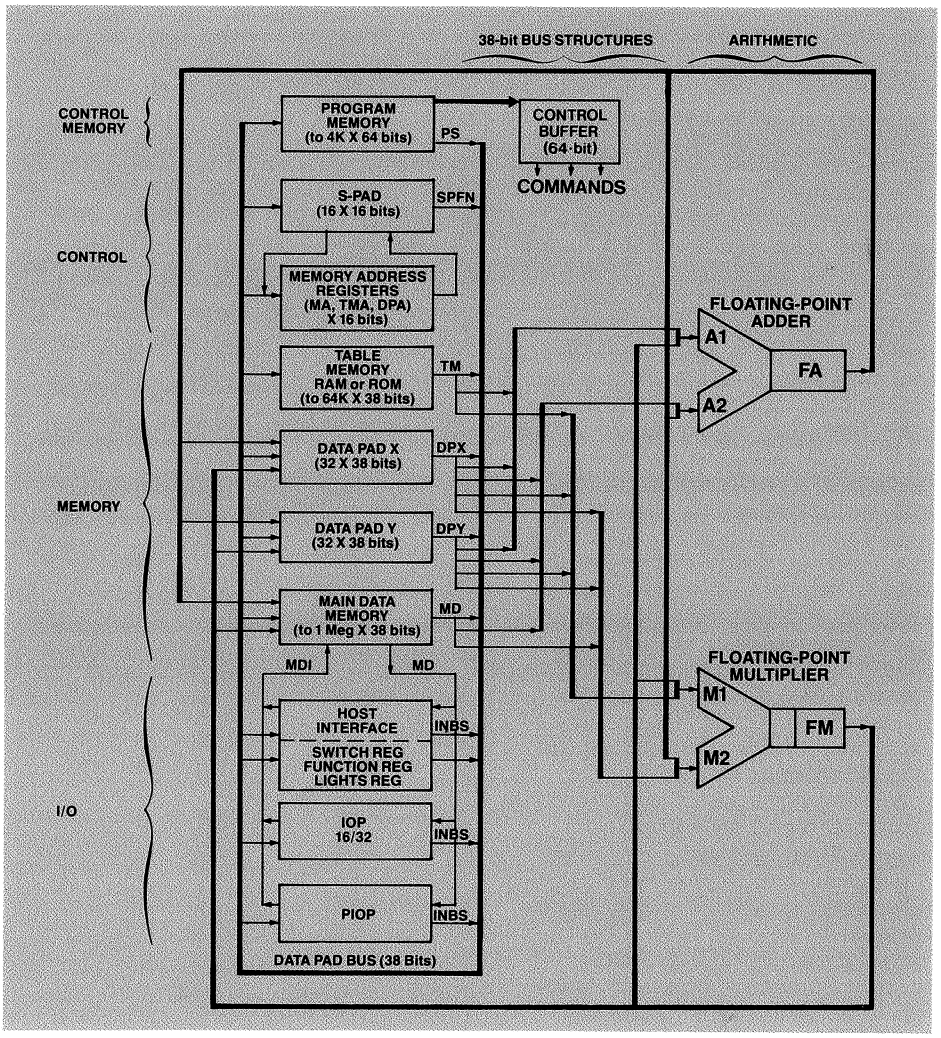
Thus, an FPS Array Processor can receive any reasonable floating-point format that is desired as the input format, convert it on-the-fly to the FPS format, process it in FPS format with minimal truncation error propagation, and then convert on-the-fly to the desired output format. This procedure allows transparent "no penalty" operation on the data.

Special architecture.

Historically, computations such as fast Fourier transforms (FFT), matrix inversions, and other repetitive calculations required excessive time (and therefore expense) in computer operations, because they were handled sequentially. To speed up these kinds of computations, processors were developed to perform a number of calculations in parallel. The goal was to achieve parallel processing of additions and multiplications simultaneously in separate logic. However, such computer circuitry has proven difficult to manufacture and program because most manufacturers have used an asynchronous multiprocessor design requiring an interaction protocol (handshaking) associated with communication between each internal microprocessor. FPS customers consider this method excessively complex and costly to control.

Independent data paths.

In addition to the carefully chosen floating-point format, array processors by FPS have **a special architecture featuring a high level of parallelism.** Seven independent data paths provide a separate bus for each floating-point arithmetic input and a separate output bus from each arithmetic element to the other elements of the system. This parallel multi-bus architecture allows great flexibility in that multiple operands and resultants can be moved simultaneously from any element to any other, minimizing access conflicts, and allowing the array processor to initiate multiple commands within each 167-nanosecond cycle. It allows the programming of specialized algorithms, such as the FFT, so they execute in times comparable to those achieved by hardwired special-purpose processors, but it also makes FPS machines well suited to less highly organized computations.



The control unit is the Array Processor's CPU. It controls all the Array Processor's elements by decoding and executing instructions from the program memory. FPS simplifies array processor control by using a unique combination of features: (1) all logic elements are controlled by one central clock (synchronous architecture); (2) all logic elements are interconnected with separate dedicated data paths to eliminate handshaking requirements, and (3) a single instruction (64-bits) with command fields controls all processing and/or memory elements. FPS Array Processors are built for simultaneous operation of all processor elements. Synchronous design assures predictable data flow, which means easier system trouble-shooting. That's good sense in system architecture.

Synchronous operation.

FPS Array Processors are **internally synchronous,** and thus have no need for internal handshaking between arithmetic units, memories, and microprocessor; data and results are available at precisely determined times. This synchronous approach has allowed a definitive simulator to be written to support easy program debugging, including single-step capability. A further bonus of the synchronous design is the total **predictability of data flow** and timing considerations, which tremendously simplifies programming by the user, and system production and test by FPS. An asynchronous processor has essentially an unlimited number of possible states, each of which should be tested — but in practice cannot be tested. In contrast, the synchronous processor has a finite number of possible states — greatly simplifying testing and enhancing reliability.

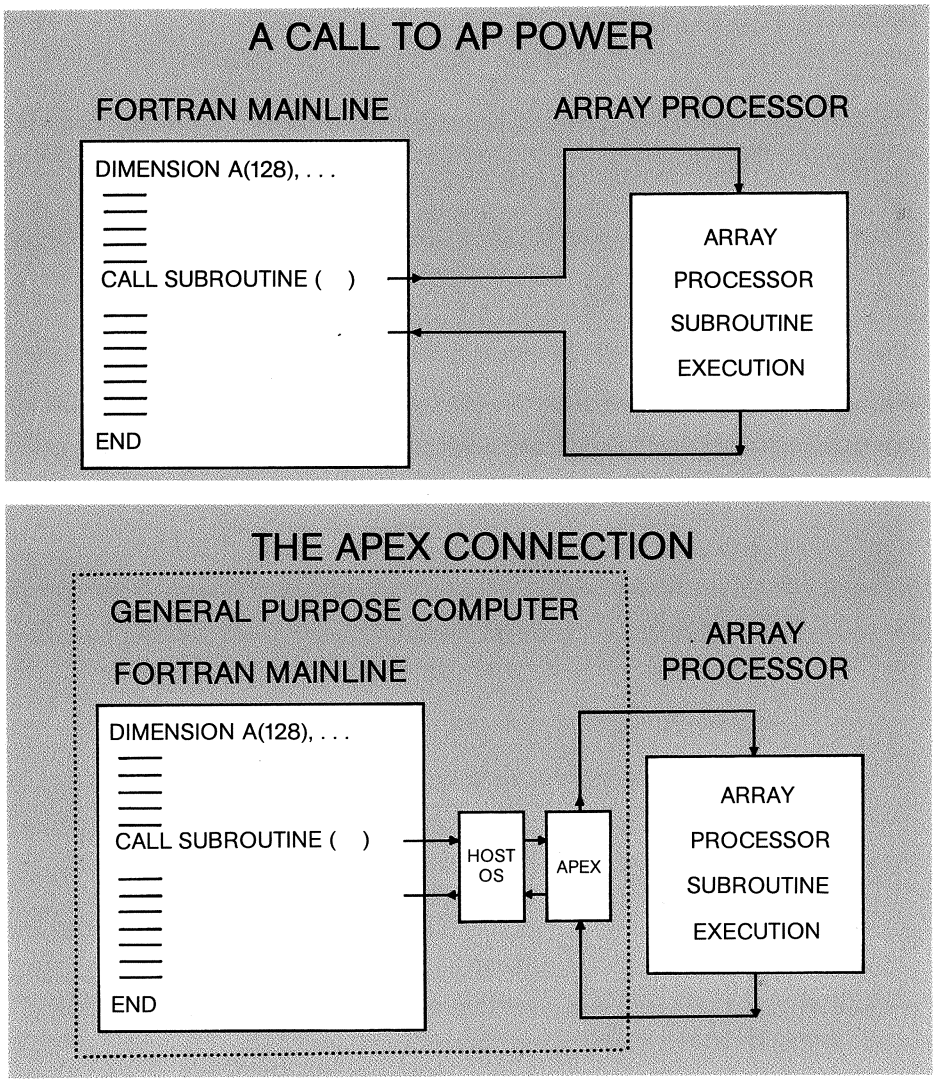
The apex connection.

Scientific computations are often accomplished on a conventional computer by a specific series of subroutines, called from a FORTRAN mainline.

When included in a computer system, **the FPS Array Processor is activated by FORTRAN calls** to computational subroutines. The subroutine calls initiate execution of programs in the Array Processor to perform the required processing.

The subroutine calls are to routines in the Array Processor Math Library and are automatically handled by the Array Processor Executive routines (APEX). APEXs have been written for all the different operating systems that FPS supports — from the relatively simple RT-11 operating system of the DEC PDP-11 minicomputer to the virtual memory operating systems associated with large computers such as IBM 370.

APEX automatically handles the passing of parameters, setting up any data transfers requested, and initiating the Array Processor's programs.



The power of FPS Array Processors can be accessed through the host's FORTRAN, which exercises FPS supplied math routines.

Simple programming.

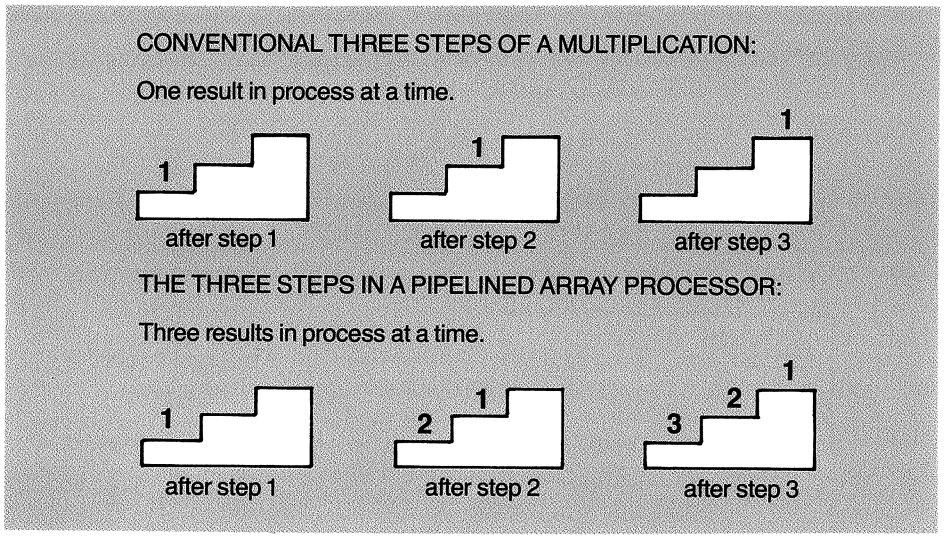
Array processors by Floating Point Systems, Inc., are supplied with a math library of more than 200 program routines, such as vector add, complex vector multiply, FFT, matrix inverse, etc., as well as a complete set of utilities for data transfer and control. These routines are callable through FORTRAN from the front-end computer.

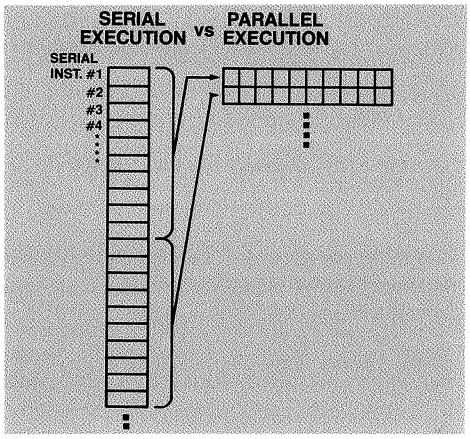
Specific application routines can be readily created for the Array Processor in one of **two ways.** The first of these is by using the **Vector Function Chainer,** which allows the programmer to automatically chain (or string) **multiple computational routines** together from the existing Math Library. Once multiple routines are chained together, the resulting computational subroutine is given its own name and added to the Math Library. This means that a very lengthy computation will be initiated in the Array Processor with a single FORTRAN call from the host computer.

The user may want to create entirely **new assembly language routines** for unique or proprietary operations, and add them to the Array Processor's already extensive Math Library. FPS provides the tools required — a Cross Assembler, a Simulator, and an Array Processor Hardware Debugger. The Simulator allows the development and testing of AP software even if your AP hardware is not yet installed. The Hardware Debugger is used for testing routines that require real time synchronization between the host and the Array Processor, and between the Array Processor and any peripherals. Added to the Math Library, these routines are now called directly from FORTRAN programs in the host computer.

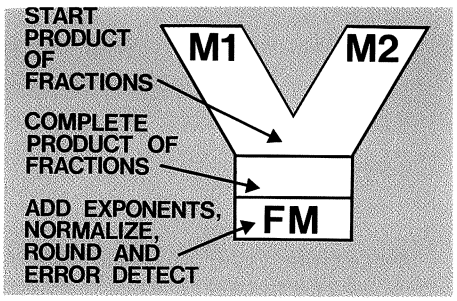
What makes a "powerful" array processor.

FPS Array Processors are powerful pipelined parallel computers. Pipelined parallel processing provides computation rates greatly exceeding conventional sequential computing rates. The floating-point arithmetic elements are pipelined for greater throughput. The two pipelines, as well as the memories and integer elements can all operate together in parallel through independent data paths. These architectural features, combined with a fast (167 ns.) instruction cycle and a large (512 K word) floating-point data memory capacity combine to give an extremely powerful, cost effective processor where high-density computation is required.





The FPS Array Processor executes computations many times faster than a conventional computer because it performs multiple operations in parallel, while conventional computers are constrained to process serially one operation at a time. In fact, the Array Processor's 64-bit wide instruction word has the capability of executing up to ten operations in a single 167 ns processor cycle.



FPS Array Processors feature 38-bit floating-point arithmetic normalized and convergently rounded to produce eight decimal digit accuracy, not just six. That's a significant advantage.

Pipelined processing.

Most arithmetic computations involve several sequential steps. A floating-point multiply, for example, might require three steps: (1) begin product of fractions, (2) add exponents and complete product, and then (3) normalization and rounding of the answer. To get the correct answer, step #1, then #2, and then #3 must be done. While #1 is being done, the hardware for steps #2 and #3 is idle, and when step #2 is being done, hardware for steps #1 and #3 is idle, and so forth. **Pipelining enhances computing throughout** by using all the stages at every step. In this case, the three steps of the multiplier hardware are in use at once. A multiply is started by doing step #1 of the process. Then without waiting for the first multiply to complete, step #1 of a second multiply is done concurrently with step #2 of the initial multiply. A third multiply is then started and now all three steps of the multiplier are in use at once.

Once the pipeline is full, a new answer is produced every time step, instead of every three steps — a three times speed improvement. In FPS Array Processors, both the floating-point adder and floating-point multiplier are pipelined. The adder can start a new add every 167 ns., and the result is available two steps later (333 ns.). The multiplier can also start a new multiply every 167 ns., with the results being available three steps (500 ns.) later. A floating-point multiply and an add every 167 nanoseconds is **12,000,000 floating-point computations per second.**

Parallel Processing.

Most conventional computers do one thing at a time. Consider the problem of summing the squares of a series of numbers. This routine would typically look like:

1. Increment data pointer
2. Fetch next data point from memory
3. Square the data value
4. Add the squared data value into the sum
5. Decrement loop counter
6. Branch to #1 if loop count non-zero

where each of these six steps is a separate instruction. Note that a floating adder or multiplier is being used in only one instruction out of six, because the conventional processor can do only one thing at a time. The conventional computer instruction word can only specify a single operation, such as multiply, add, or memory fetch, at a time. The conventional computer also has only enough data paths to support a single operation at a time.

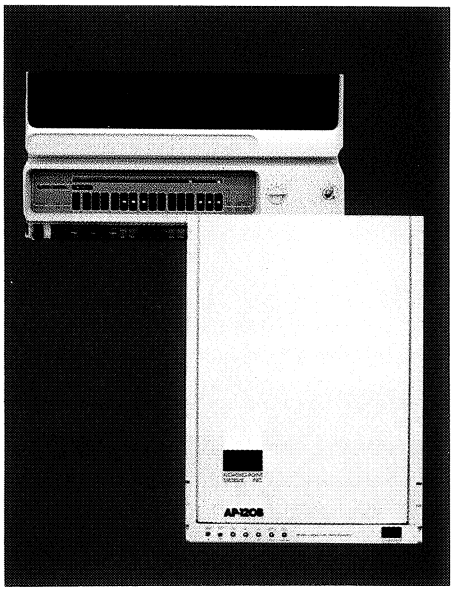
Because of their parallel pipelining, FPS Array Processors can do **all of the operations** above in a **single instruction step,** instead of six. The instruction word is wide enough so that a floating-point multiply, add, indexed memory fetch, decrement, and test can all be independently specified in a single powerful instruction.

Conventional Sequential Processor:

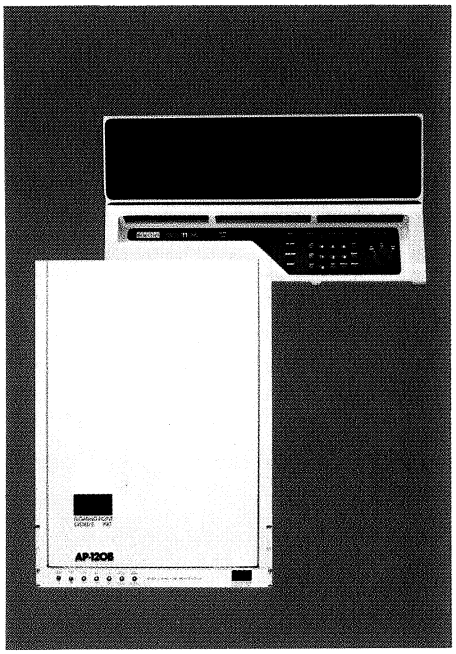
One step at a time.
Advance data pointer, then
Fetch data from memory, then
Square data, then
Add to sum, then
Decrement count, then
Branch back if not done.

Parallel Array Processor:

All the steps at once.
Advance pointer and Fetch next data and Square previous data and Add previous square to sum and Decrement count and Branch if not done.



Operating in parallel with a system computer, an FPS Array Processor may increase its computational speed and power by a hundred fold.



When an FPS Array Processor is combined with a minicomputer, the system often exceeds the capabilities of a large mainframe.

FPS array processors are interfaced to all popular computers. They are in use worldwide — performance proven.

Additional data handling capabilities

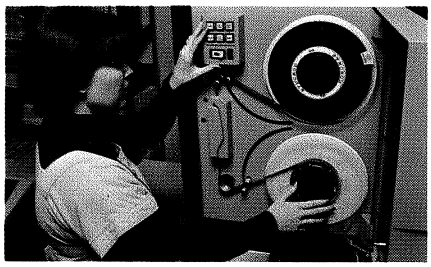
In addition to the host interface, a second path into the FPS Array Processor's Main Data Memory is available for direct data transfers to and from peripheral devices, such as A/D's and disks, without passing the data through the host system.

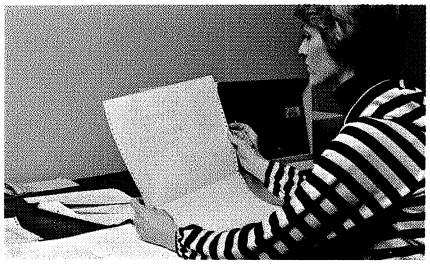
One set of input/output processors is the IOP-16/38 series. These are most appropriately used to interface analog-to-digital and digital-to-analog devices to the array processor. The IOP series allows direct memory access to the Array Processor's Main Data Memory by providing master and slave control of typical universal data bus structures. The IOP-38 (38-bit wide data transfer) is employed when direct data transfer between two Array Processors is required.

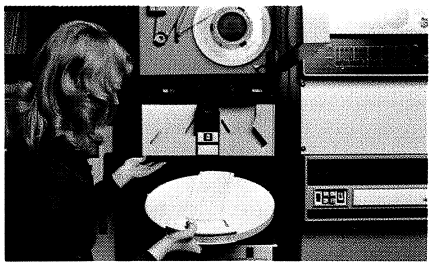
FPS has also implemented a mass storage subsystem for the Array Processor. This subsystem consists of a disk controller/formatter and either multiple 80 or 300 megabyte CDC storage module drives interfaced to the AP with a programmable I/O processor, providing the needed storage space and high speed access for applications that have large data bases. This subsystem can eliminate a considerable amount of system host I/O overhead, improving overall system performance.

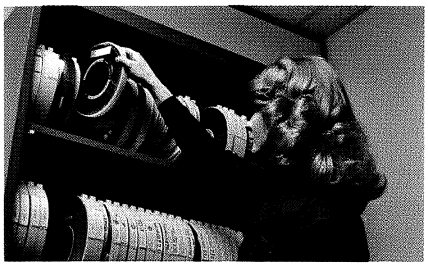
Array processors by FPS give the user the computational power of large mainframes at a fraction of the cost. They are among the most computationally powerful devices available at any price. The age of array processing is here and **Floating Point Systems, Inc., is the array processor company.**

Software









FPS offers a full complement of software that integrates the power of the FPS Array Processor into your computers operating system. FPS provides every user with the extensive FPS Math Library of more than two hundred processing routines. A program development package is also available to facilitate developing new routines, specialized for your application, and making them an integral part of the Math Library.

Convenient Access Method
Access to the powerful AP Math Library routines is made through familiar FORTRAN subroutine CALLs. All calls to the Array Processor are handled by the Array Processor Executive (APEX) software that decodes subroutine calls from your computer's FORTRAN programs and automatically passes control parameters and routines to the Array Processor for execution.

Powerful Math Library
The Array Processor Math Library (APMATH) is an extensive package of more than two hundred application and processing subroutines. These FORTRAN-callable routines, written in the FPS Array Processor assembly language, provide a multitude of vector operations, matrix arithmetic, fast Fourier transforms, and data transfer operations. An optional set of routines (AP-SIGLIB) adds specialized signal processing capability to the Math Library.

Program Development Software

FPS Array Processors are **user programmable,** allowing you to add new, special or proprietary routines to the Math Library. The Program Development Software (AP-PDS) option consists of four FORTRAN programs that are compiled on your computer system to assist in program development. These include:

1) APAL (AP Cross Assembler Program), an assembler that provides a two-pass assembly of programs written in the AP Assembly Language (APAL).
2) APLINK (AP Linker) links and relocates separate APAL object modules together into a single load module for execution.
3) APSIM (AP Simulator) simulates all aspects of the array processor on your computer. Equivalent run-time characteristics and the floating-point arithmetic (including rounding) are simulated. APSIM lets you develop programs off-line from the array processor and obtain execution times without interrupting array processor production runs.
4) APDBUG (AP Hardware Debugger) lets you interactively test new programs on the array processor. You may selectively set breakpoints, single step if desired, examine and change memory and register contents, and run program segments.

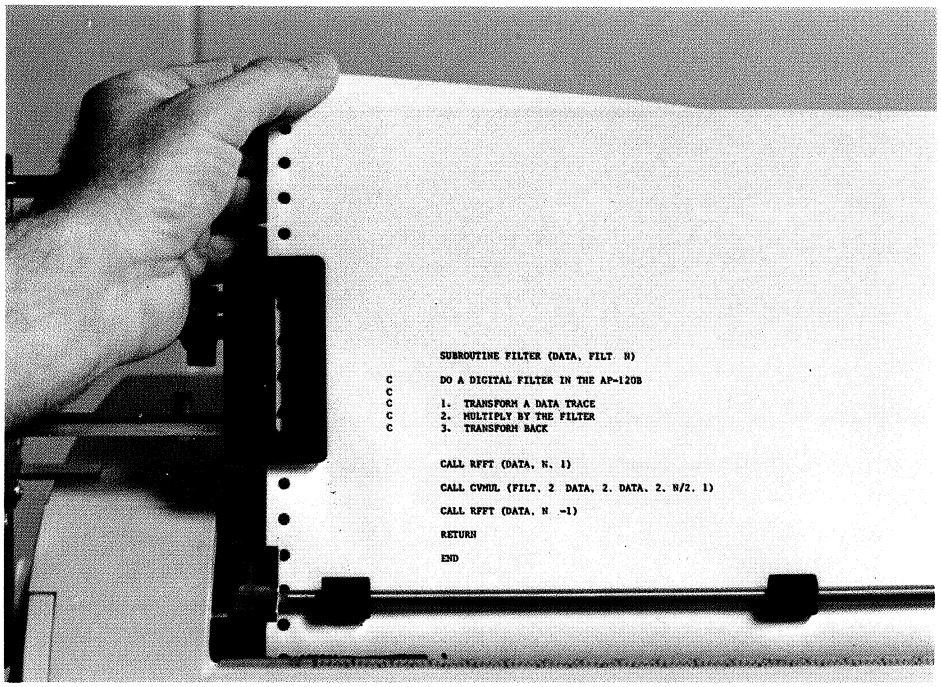
Software diagnostics.

A collection of interactive diagnostic programs test and verify operation of data paths, arithmetic elements, memory units, and the central processor. They provide you with diagnostic readout. The AP Test Programs are:

1) APTEST (AP Tester) exercises the Panel, DMA interface, and various internal registers and memories, including the main data memory.
2) APATH (AP Path Tester) tests the various internal data paths and gives board level diagnostics.
3) APARTH (AP Arithmetic Test) tests the floating-point adder, multiplier, and integer units.
4) FIFFT (Function Test) verifies the correct operation of the array processor by performing a repertoire of fast processing routines.

The Vector Function Chainer

The Vector Function Chainer automatically ties together Array Processor assembly-language routines, so that a long series of Array Processor computations can be initiated by a single FORTRAN call from your system computer, rather than separate calls for each routine. It allows the user to chain together standard FPS Math Library routines and user-created routines. While the principle benefit of the Vector Function Chainer is reduced overhead, an additional benefit is ease of use: it allows the programmer to use familiar FORTRAN syntax to create new Array Processor routines.



```
      SUBROUTINE FILTER (DATA, FILT  N)
C     DO A DIGITAL FILTER IN THE AP-120B
C
C     1.  TRANSFORM A DATA TRACE
C     2.  MULTIPLY BY THE FILTER
C     3.  TRANSFORM BACK

      CALL RFFT (DATA, N, 1)

      CALL CVMUL (FILT, 2  DATA, 2, DATA, 2, N/2, 1)

      CALL RFFT (DATA, N  -1)

      RETURN

      END
```

This example depicts a user-created filtering function that calls the Array Processor three times. The array processor routines RFFT and CVMUL can be called directly from a FORTRAN mainline, or as shown here called from an application-oriented subroutine.

Efficient programming.

Good program routines harness the power of the machine. The potential of an FPS Array Processor is revealed when you examine the length of the program loop required to accomplish a particular computation. While FPS supplies an extensive math library written **to take full advantage of the machine's capabilities,** you may write your own efficient program loops if you have special needs. For example, a lengthy but workable dot-product program loop might be 14 steps long. But since the FPS Array Processor has a highly parallel structure, each logical unit of the machine may be operated simultaneously and independently (though synchronously) for maximum speed. An array processor programmer (potentially you, as one of our customers) learns to recognize ahead of time how many cycles his loop should contain. The 14-step dot-product program can be shortened by combining instructions and overlapping memory fetches. It can actually be cut down to 3 array processor cycles! . . .

CVMUL — COMPLEX VECTOR MULTIPLY — CVMUL

PURPOSE:
To multiply the elements of two complex vectors.

FORTRAN CALL:
CALL CVMUL (A,I,B,J,C,K,N,F)

PARAMETERS:
A = Source vector base address
I = A address increment
B = Source vector base address
J = B address increment
C = Destination vector base address
K = C address increment
N = Complex element count
F = Conjugate flag,
+1 = normal complex multiply
−1 = multiply with conjugate of A

FORMULA:
$(C(mK)+iC(mK+1)) = (B(mJ)+iB(mJ+1))^*$
$(A(mI)+iA(mI+1))$ if F=1,
or $(A(mI)-iA(mI+1))$ if F=1
for m=0 to N−1

DESCRIPTION:
Multiplies N complex elements of the complex vector beginning at address B by N complex (F=1) or complex conjugate (F=−1) elements of the complex beginning at A, and stores results in the complex vector beginning at address C.

EXAMPLE:
CALL CVMUL (0,2,200,2,400,2,50,1)
Stores into AP120B locations (400,401),(402,403), . . . ,(496,497),(498,499) the complex product of the complex numbers in (0,1) and (200,201), (2,3) and (202,203), ... ,(96,97) and (296,297),(98,99) and (298,299).
The real part of the result is in locations 400,402, ... ,496,498, while the imaginary part is in locations 401,403, ... ,497,499.

EXECUTION TIME/LOOP: (us)	BEST	TYPICAL	WORST	SETUP(us)	
	1.0	1.0	1.5	2.2	(167 ns memory)
	2.0	2.0	2.5	2.0	(333 ns memory)

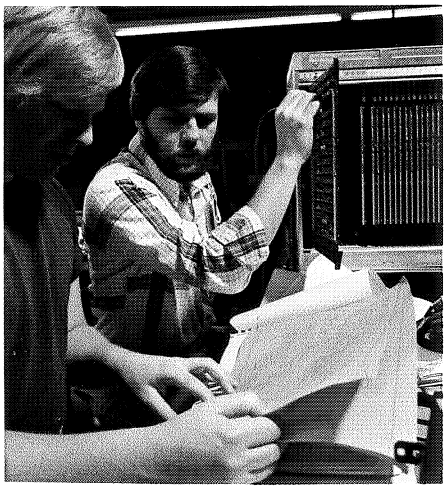
Each of the Math Library routines is fully documented for easy application by the user. Shown here is the description of CVMUL, specifying the calling parameters to be used, the analytic equation being applied, and execution time of the routine.

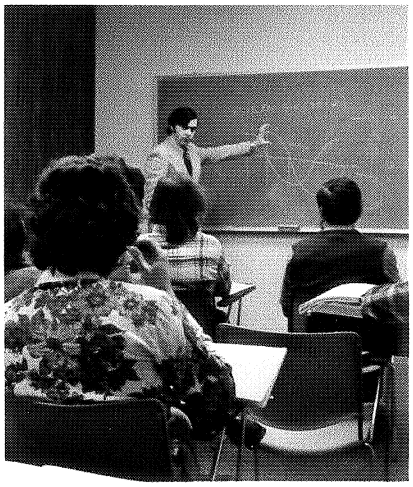
Documentation . . . everything you need, and more.

Every sale of an FPS Array Processor is backed by a complete documentation package including operation and maintenance manuals, **software source** and system schematics. Should you need further material or consultation, all you have to do is ask.

Back in the classroom again.

Floating Point Systems, Inc., offers a broad educational resource to train you or your people to the level of expertise desired. We regularly hold classes at FPS. If it is impossible for you to send your people to FPS, we can conduct classes at your location. Contact Floating Point Systems, Inc., for class schedules and rates.

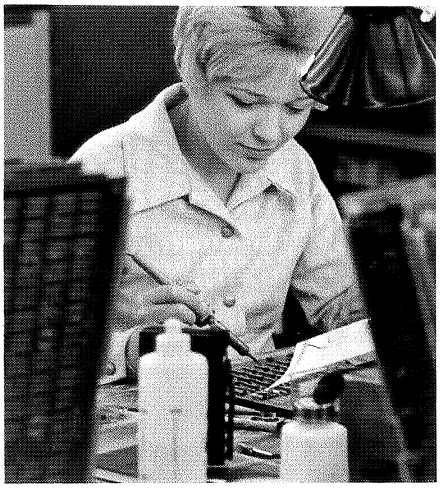




Warranty, installation, and service information.

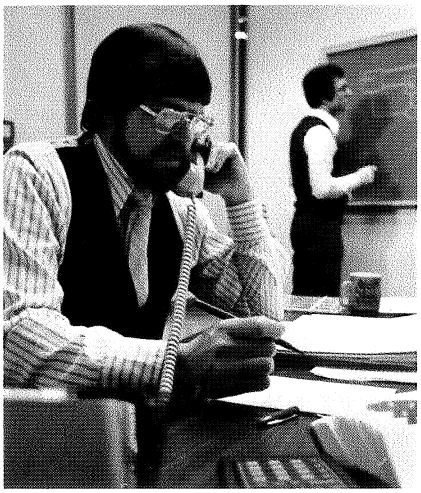
While FPS Array Processors have been in the field since mid-1975 and have demonstrated a MTBF (mean time between failure) in excess of 3500 hours, FPS stands ready to provide you with service . . . should you need it. Warranties against defects in materials and workmanship are 90 days to end users and 60 days to OEM customers, covering both parts and labor. Installations are accomplished by specially trained FPS teams, who are experienced on your computer and operating system. Service is available either on a daily rate or by service agreement. Ask your local FPS sales office for details on warranty, installation, and service. We at FPS are proud of our product, and we stand behind it.

People . . . the resource that ties it all together.



Floating Point Systems, Inc. is a high-technology computer firm with **offices worldwide** — yet it is still a people-oriented company. Founded in 1970 and located near Tektronix, we have grown to be the second-largest computer-related electronics company in the region. Today, FPS maintains equal emphasis between sales and support to end users and response to OEM customers. Company personnel supply sales support and service to end users and OEM's through a network of FPS offices that extends nationwide and overseas.

Certainly one of the catalysts for our rapid growth has been the close rapport between individuals that allows an idea to become a high-demand, high-technology product. It is this kind of people-to-people contact within FPS and between you and our company that you can depend on.



We want you, as one of our prospective customers, to give us a call. Let one of our specialists discuss your application and define how array processors can tackle your problem with surprising ease. FPS can provide specialized hardware and program routines, as well as an evaluation of your data acquisition and data processing techniques. **We continually develop hardware and software products to set the pace in array processing.** We'll bet we can provide you with a solution to your problem that is faster, more economical in both hardware and software, and more dependable.

User Group

FPS sponsors an annual User's Group Conference during which scientific papers are presented and ideas are exchanged in seminars. As an FPS Array Processor User, you are invited to attend these conferences. You also may qualify as a significant contributor to the technology of array processing. Included among the papers that have been presented at the User's Group are:

- "SEMBALNCE CALCULATIONS FOR SEISMIC VELOCITY ANALYSIS IN THE AP-120B," BY PETER BUHL, LAMONT DOHERTY GEOPHYSICAL OBSERVATORY, COLUMBIA UNIVERSITY.
- "A REAL TIME MULTI-CHANNEL DIGITAL FILTER SYSTEM," BY SAM SPARCK, TIME/DATA, DIVISION OF GEN RAD, INC.
- "THE CHEMISTRY MACHINE," BY KENT WILSON, UNIVERSITY OF CALIFORNIA AT SAN DIEGO.
- "A REAL TIME FLOATING POINT VARIABLE FRAME RATE LPC VOCODER," BY RANDY COLE, INFORMATION SCIENCES INSTITUTE, UNIVERSITY OF SOUTHERN CALIFORNIA.

A Message

The importance of providing fast floating-point hardware for the minicomputer user was the impetus for the incorporation of Floating Point Systems, Inc., in 1970. Each year since, we have introduced faster and more compact floating-point processors, spurred on by customers who in turn utilized our hardware in increasingly sophisticated applications — and more demanding hardware environments.

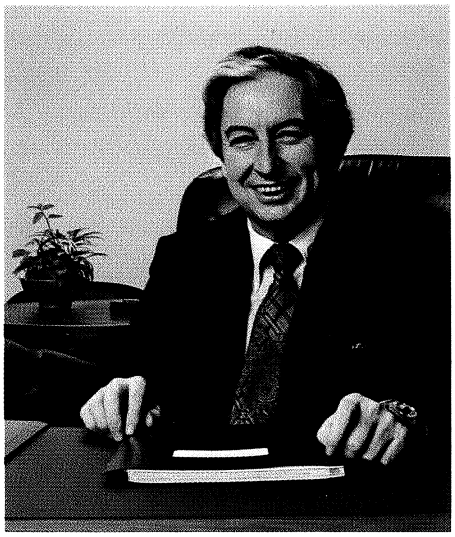
Early recognition of these new requirements led us to consider a revolutionary processor. What users needed was the very heart of a major mainframe — high-speed arithmetic power — but in a small package, and at a small price.

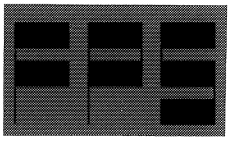
Thus, we have designed processors with general-purpose computing properties, aimed directly at high-speed algorithm execution, and made them easy to program for these tasks.

FPS Array Processors represent a uniquely advanced balance of computing precision, speed, reliability, and economy. They provide an opportunity to acquire original equipment or upgrade computational capacity while lowering investment and usage costs.

We invite all those with complex computational tasks to undertake a detailed appraisal of our array processor systems.

C. N. Winningstad
President
FLOATING POINT SYSTEMS, INC.





FLOATING POINT SYSTEMS, INC.

Call Toll Free 800-547-9677
P.O. Box 23489, Portland, OR 97223
(503) 641-3151, TLX: 360470 FLOATPOINT PTL

FLOATING POINT SYSTEMS (Canada) Ltd.
P.O. Box 11328 — Station H
Ottawa, Ontario, Canada K2H7V1
Tel: (613) 820-9608
Telex: 053-4863 FLOATPOINT OTT

INTERNATIONAL BRANCH OFFICES

FLOATING POINT SYSTEMS, S.A., LTD.
7 Rue du Marche
1204 Geneve
SWITZERLAND
Tel: 022-280453
Telex: 28870 FPSE CH

Territory Covered
Switzerland, Germany, France, Italy, Austria, Spain, Portugal, Israel, Greece, Turkey, Iran, Iraq, Algeria, Tunis, Morocco, Libya, Lebanon, Syria, Egypt, Saudi-Arabia, Kuwait, Arab Emirates, South Africa, African Countries.

FLOATING POINT SYSTEMS, S.A., Ltd.
42 Queens Road
Reading, Berkshire
ENGLAND
Tel: 0734-50181
Telex: (851) 849218 FPS UK G

Territory Covered
UNITED KINGDOM — England, Scotland, Wales, Northern Ireland
Republic of Ireland
SCANDINAVIA — Sweden, Norway, Finland, Denmark
BE-NE-LUX — Belgium, Netherlands, Luxembourg

FLOATING POINT SYSTEMS, S.A., Ltd.
D-8025 Unterhaching
Isartalstrasse 28
Munich
GERMANY
Tel: 089/619008-09
Telex: 523422 FPSGY D

Territory Covered
GERMANY

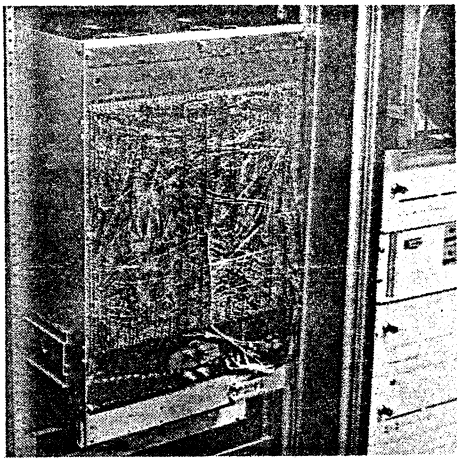
Floating Point Systems, Inc.

AP-120B

ARRAY TRANSFORM PROCESSOR

FEATURES

- Fast Floating-point arithmetic — 167ns
- Enhanced computing accuracy
- Powerful 64-bit instructions
- Multiple high-speed memories
- Numerous hardware registers
- Multiple data paths
- Overlap input/processing/output
- Efficient data widths
- Intelligent interface
- Flexible format conversion
- DMA to Peripherals
- Reliable synchronous design
- Extensive software support
- High accuracy & range



DESCRIPTION

The AP-120B floating-point array processor gives a unique combination of computational capability to mini- and maxi-computer systems. The high-speed, high-accuracy floating-point operations and programmability greatly increase system flexibility and throughput. Formatting hardware and interface logic provide full control processor and host interaction; an internal 38-bit floating-point format gives extra precision. Multiple accumulators, registers and data paths give ready access to data. Separate internal memories avoid accessing conflicts. A variety of mathematical functions and software packages allow the user to upgrade his existing computer system to handle heavy computational loads.

ARITHMETIC

The AP-120B performs floating-point arithmetic at maxi-computer speeds. Separate multiplier and adder units operate concurrently. The two-cycle pipelined adder completes a normalized, convergently rounded floating-point add (or subtract, fix, float, absolute value, logical And, Or, Equivalence) every 167ns. The three-cycle pipelined multiplier completes a normalized, convergently rounded floating-point multiply every 167ns.

Each unit detects overflow and underflow error conditions, forcing the result to the proper signed maximum value or zero, and setting the appropriate error flag.

+time = 334nS, ± 501nS

ACCURACY

The floating-point adder and multiplier both contain special hardware to provide an extra cushion of accuracy for each computational step.

The adder has three extra bits (equal to 28 extra bits for normalized arguments) to preserve information shifted right during exponent alignment for use later during normalization and convergent rounding. The binary floating-point format further minimizes loss of resolution.

The multiplier computes the entire 56-bit fractional product, and then normalizes and convergently rounds to 28-bits.

Convergent rounding hardware in both arithmetic units rounds the normalized result up only when the remainder is greater than one half of the least significant bit. This causes the cumulative rounding error bias to converge toward zero.

INSTRUCTIONS

A single AP-120B instruction handles operations that would require several successive instructions on most computers. Each 64-bit instruction is divided into 10 powerful command fields, allowing a single instruction to perform a variety of tasks:

Floating-point add
Floating-point multiply
Fetch or store from Data memory
Read and store accumulators (two reads and two writes)
Conditional branching
Fetch from Table memory
Index register arithmetic

All transfers, computations, and memory accesses occur synchronously. No special programmed testing is needed to insure proper execution.

The AP-120B has three separate high-speed memories, each with dedicated controller logic. Multiple memories eliminate accessing conflicts, allow an optimal word length and cycle speed for each, and provide flexibility in configuring for specific applications.

Program Memory: 64-bit word size; 50ns cycle bipolar memory in 256-word increments. Addressable to 4K words.

Data Memory: 38-bits word size; either 167ns interleaved cycle MOS memory in 4K word increments, or 333ns interleaved cycle MOS memory in 8K word increments. Two parity bits are optional. Addressable to 1 Megawords.

Table Memory: 38-bit word size, 167ns cycle bipolar ROM in 512 word increments. Addressable to 64K words. Optional 167ns RAM table memory available in 1024 word increments up to 64K.

ACCUMULATORS AND REGISTERS

The AP-120B has 64 floating-point accumulators. Four accumulators are available during any instruction cycle; two as argument sources and two as result destinations.

The 16 integer index registers permit address calculations and loop overhead to occur concurrently with floating-point arithmetic. Data arrays may be indexed in either true or bit-reversed order.

DATA PATHS

Four separate 38-bit wide data paths into the floating-point adder and multiplier allow an uninterrupted flow of arguments for computation. Three additional data paths concurrently channel results back into accumulators or memory. This combination of multiple data paths allows sustained computational throughput, and avoids intermediate data moves. DMA cycle stealing allows overlapped data input, arithmetic processing, and data output.

DATA WIDTHS

Three distinct data widths are used to fit each particular data type: instructions, floating-point numbers and integers.

The 64-bit instruction word permits up to 10 different instructions to be programmed concurrently in a single 167ns processor cycle.

The 38-bit floating-point data word gives both extra computing accuracy and increased dynamic range over traditional 32-bit word lengths (8.1 vs 6.0 decimal digits, with a dynamic range of over $10^{\pm 153}$).

The 16-bit integer word allows direct memory addressing to 65,536 words; and to 1 million words with memory bank selection.

INTERFACE

An intelligent interface coordinates interaction between the AP-120B processor and the host computer.

The AP-120B electronic front panel gives the host computer access inside the AP-120B. The internal memories and registers may be examined and modified. A hardware break-point can be set to stop AP-120B execution at a selected program location or data address.

Programs may be single stepped for debugging purposes, with execution identical to free-running programs.

The interface provides four data transfer combinations between the two processors: host DMA to AP-120B DMA; Host DMA to AP-120B programmed 1/0; host programmed 1/0 to AP-120B DMA; and host programmed 1/0 to AP-120B programmed 1/0. Either the AP-120B or the host computer can control data transfers.

Rowland.
If anyone is interested I have got some more information
David Eglin

FORMAT CONVERSION

The formatting hardware in the AP-120B interface converts data "on-the-fly" during transfers between the AP-120B and host or peripherals. Four standard formats are automatically handled and many options are available.

Host Computer Format	to AP-120B Format
16-bit integers	– 38-bit unnormalized floating point
32-bit integers	– 32-bit integers
IBM 360 floating point	– 38-bit normalized floating point
Host floating point	– 38-bit normalized floating point

On return of results to the host, any floating-point numbers that would be out of range for the target format are detected and forced to the proper signed maximum or zero, and indicator bits set.

PERIPHERALS

A separate programmed 1/0 and direct memory access bus is provided for peripheral devices (A/D converters, disks, tape drives, etc.) to be interfaced directly to the AP-120B. Up to 256 1/0 device addresses can be accommodated.

RELIABILITY

A single master clock synchronizes processor actions in the AP-120B. There is no uncertainty, caused by inter-element "hand-shaking", not variable phasing delays.

Timing allowances are designed to worst case temperature and voltage operating specifications — a full 50% safety margin in every cycle.

Units are tested as a functioning system at extremes of heat and supply voltage: a minimum of eight successful hours at high temperature, high voltage and again at low temperature, low voltage.

The high-speed Schottky MSI and LSI TTL logic is pre conditioned and parametrically tested to eliminate "infant mortality".

COMPACT PACKAGING

The AP-120B is arithmetic power in a compact package. The chassis mounts in standard 19' rack and occupies 24½" of space, including a high-efficiency switching regulated power supply. Rugged mechanical design and moderate card size (10" x 15") maximize system integrity in the field.

SOFTWARE SUPPORT

An extensive package of library functions, vector and matrix subroutines, and signal processing algorithms is supplied with the AP-120B; all callable from the host computer's FOR-TRAN programs. A cross assembler, program linker, on-line debugger (using the electronic front panel) and software simulator aid in new program development.

ALGORITHM EXECUTION TIMES*

Multiply-add: 167ns
Complex multiply: 667ns

Algorithm	167ns memory	333ns memory
1024 pt real FFT	2.9 ms	4.1 ms
1024 pt complex FFT	5.5 ms	6.9 ms
8192 pt real FFT	28.4 ms	37.7 ms
1024 pt vector elem. by elem. mult.	0.52 ms	1.05 ms
1024 x 32 pt convolution	6.0 ms	6.5 ms
512 x 512 real 2 DIM FFT	1.55 sec	2.2 sec
1000 element vector square root	1.85 ms	1.85 ms

*Established by benchmarks at user installations.

SPECIFICATIONS

ARITHMETIC

Floating-point width: 38-bits
Floating-point adds: every 167ns, 333ns for completion
Floating-point multiples: every 167ns, 500ns for completion
Integer width: 16-bits
Integer arithmetic: 167ns

MEMORY

Cycle: Program Memory, (bipolar RAM); Data Memory, 167ns interleaved or 333ns interleaved (MOS); Table Memory, 167ns (bipolar ROM).

Sizes: Program Memory, increments of 256 words to 4K words; Data Memory, increments of 2K words (167ns) or 8K words (333ns) to 1 million words; Table Memory, increments of 512 words to 64K words.

Word Sizes: Program Memory, 64 bits,
Data Memory, 38-bits, plus two optional parity bits
Table Memory, 38 bits.

REGISTERS

Floating-point Accumulators: 64
Index registers: 16
Subroutine return stack: 16

INPUT/OUTPUT

Direct memory access rate: Up to 3 million 38-bit words per second, depending upon the host computer.

Input/Output System, 38-bit word length, 256 devices addressable. An ancillary DMA port allows the AP-120B to communicate with other peripherals, such as disc memory A/D and D/A converters etc.

ELECTRICAL

Power service: 105/125 at 20 amps or 188/228 Vac, or 210/250 Vac at 10 amps. 50/60 Hz. 50/400 Hz optional.
Power Dissipation with 512 words of Program Memory, 16K Data Memory, and 2K Table Memory; less than 1200 watts.

ENVIRONMENTAL

Temperature Range: 10 to 40°C operating

Relative humidity range: 0-90%
Ventilation: 8 push-pull fans.

PHYSICAL

Dimensions: 24½" H, 19" W, 20"-25" D
Weight: 88 pounds including power supplies

Phone: (503) 620-1980 TELEX 360470 FLOATPOINT PTL

FLOATING POINT SYSTEMS, INCORPORATED
10520 S.W. CASCADE BLVD. PORTLAND, OREGON 97223

Form 7244
Rev. 1 May 76

COMPARISON OF SELECTED ARRAY PROCESSOR ARCHITECTURES

Examination of synchronous parallel pipelined and asynchronous parallel array processor architectures points out differing hardware/software concepts, memory utilization, input/output flexibility, and processing capabilities

Stephen P. Hufnagel The University of Texas, Austin, Texas

Large general purpose mainframe computers, while capable of very high throughput rates, are too expensive for performing dedicated scientific computations. Although lower in cost, minicomputers have limiting processing speeds for handling repetitive numerically intensive calculations. Consequently, requirements for cost-effective solutions of iterative arithmetic on large arrays of data have led to the design and development of the add-on programmable array processor. In conjunction with a frontend host minicomputer, the array processor provides substantially increased throughput, precision, and dynamic range, under program control, for accommodating a wide range of digital signal processing applications. These applications, such as radar or sonar detection, speech analysis, data reduction, and image enhancement, usually involve exacting solutions of a defined set of problems, employing either fast Fourier transform, matrix operation, correlation, or convolution algorithms.

Combining an array processor with a host minicomputer allows each to perform individually and optimally. The host provides overall system control, directing the flow of data and instructions between input/output (I/O) peripherals and the array processor. High speed complex calculations are executed by the array processor, more than 100 times faster than the host could perform them.

For effective interface with a host minicomputer, an array processor should encompass (a) several specialized processors functioning in parallel, (b) floating point data format, (c) programmability, and (d) direct connections to external devices. The latter significantly reduces activity in host memory since only processed results are stored there. Software adaptability allows program deletions, changes, or modifications to existing algorithms, and enables new routines to meet altered application requirements. Rounding hardware in the floating point arithmetic units, compatible with fixed point formats, promotes improved accuracy. Multiple, parallel processors, under effective control, yield high speed operation.

Comparison of the architectural design of two programmable array processors reveals capabilities associated with each. Both processors perform high speed, floating point calculations as an attached peripheral device to a host minicomputer at an estimated processing rate exceeding 10M instructions/s. Unit A is the Floating Point Systems AP-120B; this unit is a synchronous, parallel pipelined, horizontally microprogrammable machine with a 167-ns cycle time, a 38-bit floating point format, a 64-bit program control word, 64 floating point accumulators, and 16 integer index registers. Unit B is the CSP MAP-300; this unit is an asynchronous, parallel, vertically microprogrammable

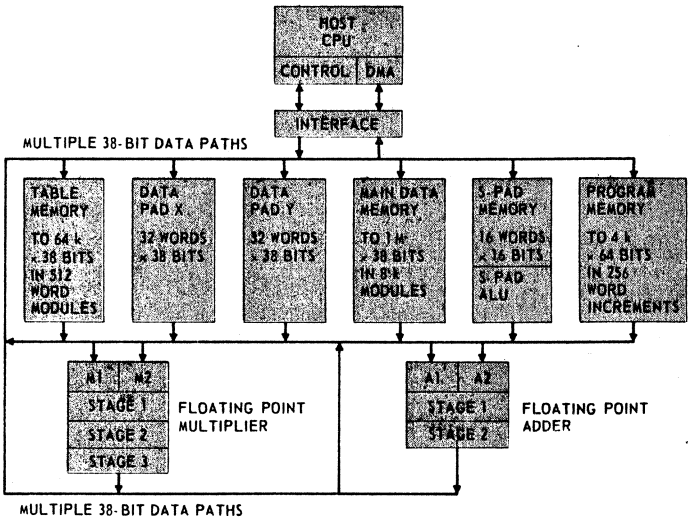


Fig 1 AP-120B array processor (unit A). Synchronous, parallel pipelined, horizontally microprogrammable machine has separate input bus and output bus for each floating point arithmetic element to any other element of system. Parallel multibus architecture allows flexibility in that multiple operands and resultants can be moved simultaneously from element to element

machine with a 70-ns cycle time, a 32-bit floating point format, a 16-bit program control word, 32 floating point accumulators, 8 integer index registers, and 3 parallel interprocessor memory buses.

Synchronous Parallel Pipelined Architecture

Significant design modules and the interconnecting data flow bus structure of unit A are shown in Fig 1. The host computer, which is typically a 16-bit minicomputer, serves the functions of program loading, I/O device setup and sequencing, possible data source, and possible pre- and/or post-data processor for unit A. This host is mandatory since unit A has only a primitive I/O capability to set up a direct memory access (DMA) transfer and to determine whether the transfer has been completed.

No interrupt handling hardware is available. An interface routes and directly ties the host computer to multiple 38-bit data paths. These paths transfer data between remaining blocks in the diagram. Any block is able to send data to, or receive data from, any other block with the exception of S-pad memory and S-pad arithmetic logic unit (ALU). S-pad memory performs integer address indexing and loop counting, and contains sixteen 16-bit directly addressable registers. The contents of these registers are modified to provide address routing registers for table memory, data pad X and data pad Y registers, and main data memory.

The floating point adder is a 2-stage pipeline adder with two buffer registers at the input. This pipeline design yields an effective floating point add rate that is equivalent to the 167-ns cycle time of the array processor, once the pipeline is full. Alternately, two cycles are required for the result of a particular add to pass through the pipeline before the sum is usable in any further calculation.

A 3-stage pipeline, the floating point multiplier has two buffer input registers and must sequence through three stages before a particular result is available. Like the floating point adder, the effective floating point multiply time is one machine cycle time of 167 ns.

Two independently addressable banks of 32 floating point registers comprise data pad X and data pad Y. Each word is 38 bits long, and any register can be accessed in a single machine cycle. Additionally, a simultaneous read and write is possible within each data pad during the same machine cycle. This parallel register load and store operation allows storage of adder or multiplier results, or of data obtained from memory in the register, and fetching of new register data for input to adder, multiplier, or memory to occur simultaneously.

Control of unit A is achieved by 10 parallel opcode fields contained in the 64-bit program control word (Fig 2). All memory accesses, register transfers, and computations occur synchronously and in parallel. This sequencing allows for the simultaneous execution of the floating point adder, floating point multiplier, a fetch or store from data memory, two reads and two stores of accumulators, a conditional branch test, a fetch from table memory, and index register integer arithmetic to facilitate data memory, table memory, and register addressing.

Program memory has a 64-bit wide word and a 50-ns cycle time to support the processor. For any type of complex logical operations in memory, register address decoding, demultiplexing, or for any conditional logic based upon the data themselves, efficiency of this machine approaches that of a serial microencoded machine. The pipeline design of the adder and multiplier further diminishes the machine's efficiency when logical operations are based on the data themselves. Before a test

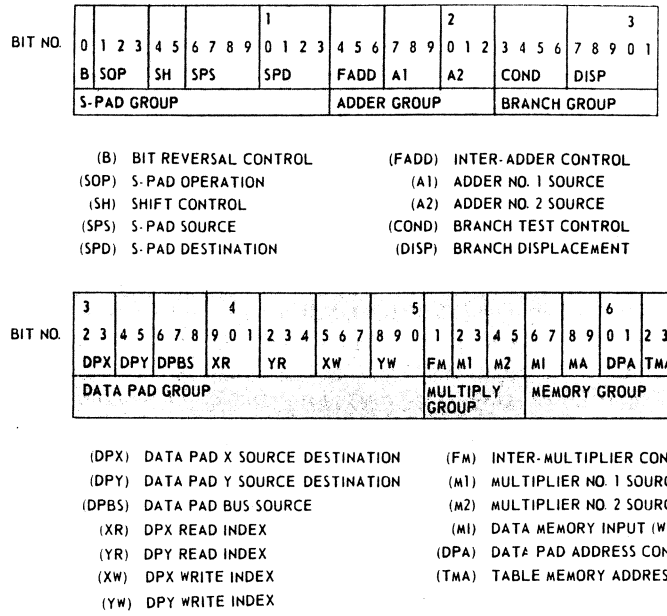
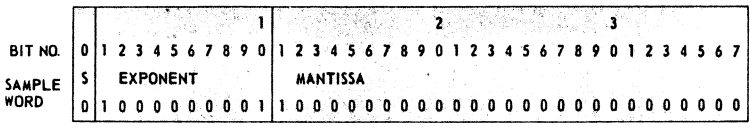


Fig 2 64-bit program control word. Control of Unit A is achieved by 10 parallel operation groups. S-pad field performs integer address indexing, loop counting, and control functions; adder group controls two floating point adders; branch group performs data dependent conditional testing; data pad group controls two 32-word floating point fast accumulator blocks, data pad X and data pad Y; multiply group forms product of two multiplier input registers; and memory group controls data memory and table memory



THE SAMPLE WORD IS 1.0. THE SIGN IS POSITIVE, HENCE S = 0; THE EXPONENT IS 1 BIASED BY 512_{10}, WHICH IS 1001_8; AND THE FRACTIONAL MANTISSA IS A NORMALIZED NUMBER WITH THE MOST SIGNIFICANT BIT SET AND ALL OTHER BITS EQUAL TO ZERO.

Fig 3 Unit A data word format. 38-bit floating point data word starts with sign bit (S); next, 10 bits of exponent biased by 512; and then, 27 bits of fractional mantissa. Data word gives both extra computing accuracy and increased dynamic range over traditional 32-bit word lengths. Approximate dynamic range of $10^{\pm 163}$ is available

can be made, data must completely pass through the pipeline, which is two cycles for add and three for multiply. This normally prohibits the pipeline from being utilized for any other function in the interim. Hence, the effective 167-ns, parallel, floating point multiply and add times may be reduced to serial 835 ns and 668 ns, respectively, to facilitate conditional logic operations on the data. Sequencer design allows maximum utilization of available arithmetic processing resources by inefficiently utilizing program control memory.

Program code becomes wasteful in the limited 4k words of high cost program storage. For each logical condition, a partially redundant program loop must be written to attempt to utilize as many processing resources in parallel as possible. However, this inefficient use of program storage does allow optimization of processing speed.

In a perfect processing environment, the sequencer would be able to control 12M floating point calculations/s. Although this number provides a performance measurement criterion, it is unlikely that it represents a possible programming goal for any realistic application to achieve.

Data format selected for unit A is illustrated in Fig 3 as a binary, 38-bit floating point word with a 10-bit exponent, biased by 512, and a 28-bit 2's complement fractional mantissa. It is anticipated that only 32 bits would be input from, or returned to, the host computer. The extra six bits of the internal floating point format are considered guard bits to minimize computational error.

The memory hierarchy of unit A entails use of high speed bipolar read/write program memory (RAM), metal-oxide semiconductor (MOS) data memory, and bipolar read-only memory (ROM) table memory. The advantage of each technology has been used to satisfy the relative needs of program control, data processing, and bulk data storage.

Program storage memory, which has a 64-bit word size, a 50-ns cycle time, and is addressable to 4k words in 256-word increments, is the highest performance, but most expensive memory, in the system. On the other hand, data storage memory represents

the tradeoffs required to provide bulk storage at a reduced cost.

Two types of data memory are available. They both have a 38-bit word size and are constructed of MOS integrated circuits. Data memory is addressable to 1M words with either 167- or 333-ns interleaved cycle time. An obvious tradeoff between processing requirements and cost can be made between the two memory speeds. In complex iterative algorithms, the two high speed 32-word data pad registers should prove adequate to prevent the slower memory cycle time from becoming a limiting factor in processing ability.

Table memory consists of 38-bit word, 167-ns cycle time bipolar ROM or RAM, in 512-word increments, addressable to 64k words. Table memory is used as constant storage and, for a fixed algorithm application, low cost ROM is adequate. Additional flexibility is obtained by using either a mixture of ROM and RAM table memory or high cost RAM table memory only.

The I/O system for unit A is a 3M, 38-bit word/s, direct memory access (DMA) common bus to either program or data memory. All inputs and outputs are on a memory cycle steal basis, and up to 256 devices are addressable.

Under program control, unit A is able to load program memory with an alternate program code provided by the host computer. This capability helps compensate for the limited size of this memory in applications where programs can be partitioned into independent sets of code and where the sequence of events allows sufficient time to replace code in program memory.

Data memory is also updated by DMA transfer. One limitation of this approach is the necessary cycle-steal sequence, which will stop all processing in the array processor for one memory cycle if a memory reference is being made simultaneously with the input or output transfer of a peripheral device or the host computer.

In summary, the architecture of unit A is designed to speed up data processing limitations of the traditional computer ALU by a synchronous parallelism of operations. This parallel multibus architecture provides high processing speed because multiple operations can be performed and resultants can be moved simultaneously between elements. The synchronous approach provides total predictability of data flow and timing considerations, which simplifies programming. However, the parallelism of operation and the pipeline multiplier and adder lose much of their speed advantage in an algorithm requiring logical decisions based upon data value. In addition, logical operations require inefficient use of expensive program memory.

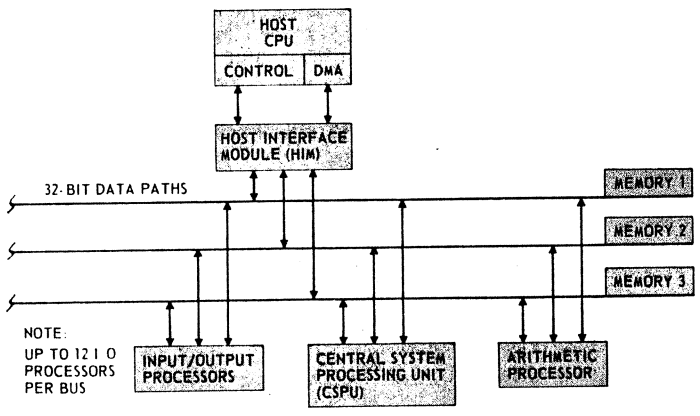
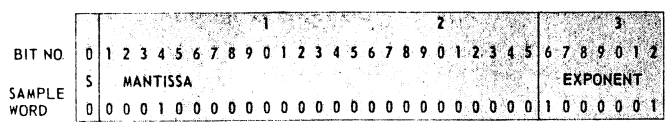


Fig 4 MAP-300 array processor (Unit B). Machine has individually programmable asynchronous multiprocessors, which require an interaction protocol associated with communications between each processor. Separate I/O processor provides exceptional overall I/O characteristics by handling DMA block transfers between external devices and three memories, or PIO transfers of packed or unpacked and multiplexed or demultiplexed data at rates up to 8 MHz and 1.8 MHz, respectively



THE SAMPLE WORD IS 1.0. THE SIGN IS POSITIVE, HENCE S = 0; THE EXPONENT IS 1 BIASED BY 64_{10}, WHICH IS 41_{16}; AND THE FRACTIONAL MANTISSA IS A HEXADECIMAL NORMALIZED NUMBER WITH THE FOURTH MOST SIGNIFICANT BIT SET AND ALL OTHER BITS EQUAL TO ZERO.

Fig 5 Unit B data format. IBM compatible floating point word has sign bit (S), 7-bit hexadecimal exponent encoded in excess-64 code, and 24-bit hexadecimal mantissa. Mantissa limits guaranteed precision to 21 bits, and exponent provides dynamic range of approximately $10^{\pm 70}$

Asynchronous Parallel Architecture

Control of unit B is achieved by an individually programmable asynchronous multiprocessor design requiring an interaction protocol (handshaking) associated with communications between each microprocessor. Unit B consists of the central system processor unit (CSPU), arithmetic processor, memories, host interface, and I/O processors (see Fig 4).

The data format selected for unit B (see Fig 5) is an IBM floating point format, which consists of a 7-bit hexadecimal exponent, a 24-bit hexadecimal mantissa, and a sign bit. However, a normalized hexadecimal positive number can still have as many as three leading zeros, which results in reduced precision of the data format over a binary format of equivalent size.

Serving as executive processor in the multiprocessor system, the CSPU communicates with each individual processor through three common memory buses and through a programmable priority interrupt system. The CSPU initiates processing sequences by entering needed programs into the arithmetic processor and the address controller of the arithmetic processor, and by starting operations as tasks become available from the host computer. It generally controls data flow in the system, including control of the I/O processors. The CSPU has a 16-bit fixed point arithmetic unit with a 125-ns cycle time, an 8-register accumulator file, and a 192-level priority interrupt network.

The arithmetic processor can be divided into functional units whose operations are interdependent. A pair of arithmetic processing units is parallel programmed from 128 words by 32 bits of read/write memory. These two processors each have a floating point adder, a floating point multiplier, 16 floating point accumulators, and they share I/O first-in first-out (FIFO) queues (Fig 6).

These queues communicate to the three memory buses and are routed and addressed by the address queues provided by the independently programmable arithmetic address processor. The single operation field of the arithmetic control program word (Fig 7) allows for a sequential multiply, add, data transfer, conditional jump, or unconditional jump. Operations take from one to six subcycles, where a subcycle is defined as 70 ns, and the program sequencer allows parallel utilization of resources. A program step wait is generated if a resource is requested when the operation necessary to provide the resource is not completed.

The arithmetic address processor runs parallel to the arithmetic processor and has a repertoire of integer instructions to allow calculations of multiplex and

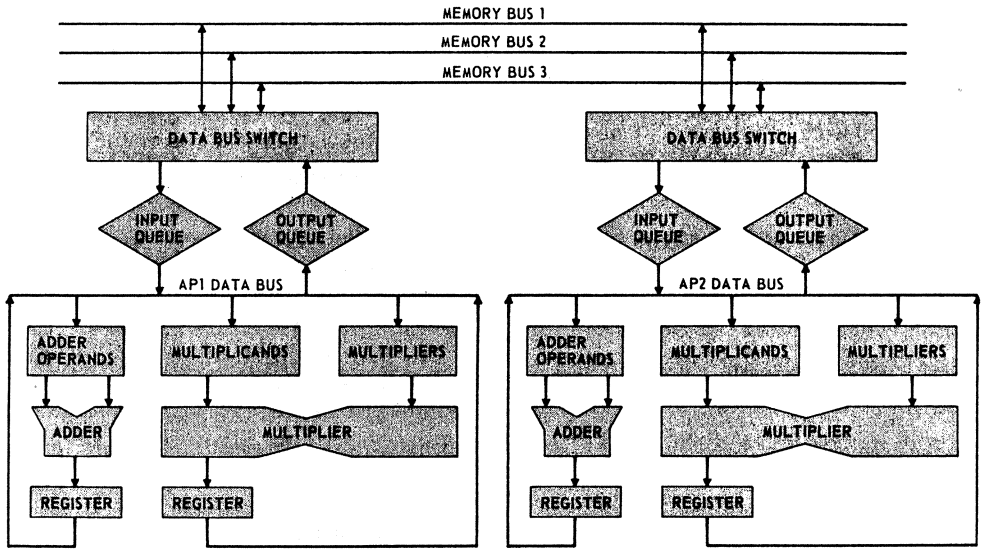


Fig 6 Dual arithmetic processor. Each processor has pair of independently programmed parallel adders and multipliers that feed through input and output queues to memory buses. Parallel asynchronous multiprocessor design increases overall speed, processing flexibility, and interprocessor control

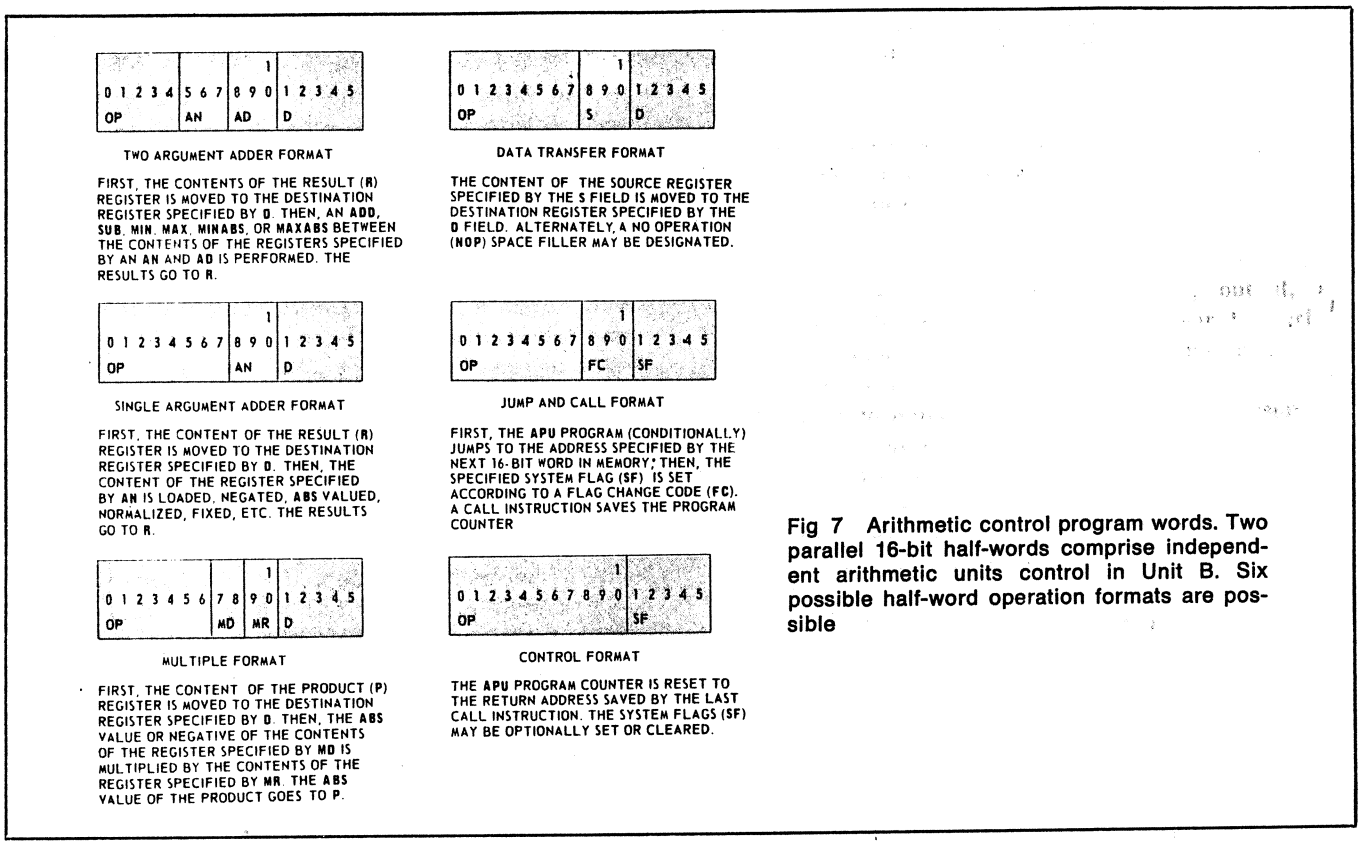


Fig 7 Arithmetic control program words. Two parallel 16-bit half-words comprise independent arithmetic units control in Unit B. Six possible half-word operation formats are possible

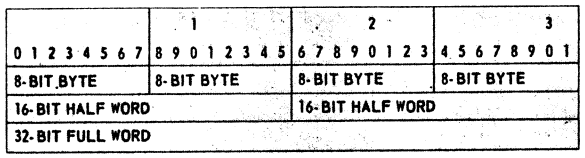


Fig 8 Versatile memory address ability. Unit B efficiently utilizes 32-bit main data storage supporting byte, half-word, and full-word addressing. Floating point numbers may be addressed as 32- or 16-bit words while integer numbers are addressed as 16- or 8-bit words

demultiplex address patterns for the three memory address buses. Input and output addresses are fed into separate FIFO queues that work in conjunction with the input and output data queues of the arithmetic processor sequencer. The arithmetic address processor has 128 words, each containing 25 bits of programmable memory that are writable by the CSPU.

Because of parallel arithmetic units, small program memory, and asynchronous address and processor I/O structures, complex logical decisions based upon data values would be inefficiently executed in this processor. Alternately, predetermined data multiplexing and demultiplexing are possible between the processor and memory as a result of the arithmetic address processor.

Unit B is capable of performing 4.75M multiplies/s plus 9.5M adds/s nested behind the multiplies. This capability provides a maximum possible throughput of 14.25M floating point arithmetic calculations/s. Realistically, this criterion is not an attainable programming goal for normal applications.

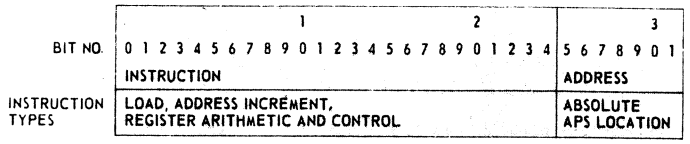
The memory hierarchy of unit B entails use of 125-ns cycle time bipolar RAM and 500-ns cycle time MOS memory. All program memory is high speed RAM, while main memory for data storage and program storage can be a mixture of both RAM and MOS memory.

Main data storage and CSPU control program memory use 32-bit words that are addressable in 8-bit bytes, 16-bit half-words, and 32-bit full-word increments (Fig 8). This addressability facilitates efficient memory utilization.

Memory architecture is such that each of the three independent memory buses is capable of addressing 64k words. Since multiple memories may be employed and since each bus operates independently, each subprocessor can access a memory simultaneously on a cycle-steal basis, or can overlap input, output, and processing. This allows each subprocessor to perform at its maximum rate. Memories are multiported with up to 16 ports each.

Program memory for the arithmetic processor is a 128-word, 32-bit/word RAM. Each word is partitioned into half-words (Fig 9) to allow parallel control of the two multiplier and adder pairs that make up the arithmetic processor.

The arithmetic addresser section memory of the arithmetic processor has a 128-word, 25-bit/word RAM. Under direction of the CSPU, the arithmetic address section programs are loaded from main memory. The 32-bit words from main memory consist of a 7-bit address and a 25-bit instruction (Fig 10). The 7-bit



BIT NO.	1 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4	2 5 6 7 8 9 0 1
	INSTRUCTION	ADDRESS
INSTRUCTION TYPES	LOAD, ADDRESS INCREMENT, REGISTER ARITHMETIC AND CONTROL	ABSOLUTE APS LOCATION

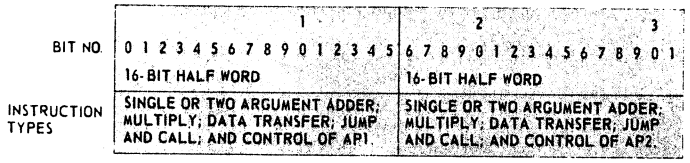
LOAD INSTRUCTION: LOADS THE CONTENTS OF THE DESIGNATED OPERAND REGISTER WITH A SPECIFIED 17-BIT ABSOLUTE MAP MEMORY ADDRESS.

ADDRESS INCREMENT INSTRUCTION: ARITHMETICALLY COMBINES THE SPECIFIED INCREMENT WITH THE CONTENTS OF THE DESIGNATED OPERAND REGISTER.

REGISTER ARITHMETIC INSTRUCTION: ARITHMETICALLY COMBINES THE SPECIFIED INCREMENT WITH THE CONTENTS OF THE DESIGNATED OPERAND REGISTER.

CONTROL INSTRUCTIONS: A (CONDITIONAL) JUMP IS SPECIFIED.

Fig 9 Arithmetic processor program memory. Program memory of Unit B arithmetic processor is partitioned into half-words to allow parallel control of two multiplier and adder pairs



BIT NO.	1 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5	2 3 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
	16-BIT HALF WORD	16-BIT HALF WORD
INSTRUCTION TYPES	SINGLE OR TWO ARGUMENT ADDER; MULTIPLY; DATA TRANSFER; JUMP AND CALL; AND CONTROL OF AP1	SINGLE OR TWO ARGUMENT ADDER; MULTIPLY; DATA TRANSFER; JUMP AND CALL; AND CONTROL OF AP2

SINGLE ARGUMENT ADDER INSTRUCTION: PERFORMS ONE OF 13 ARITHMETIC OPERATIONS ON THE CONTENTS OF AN OPERAND REGISTER TO TAKE THE ABSOLUTE VALUE, APPROXIMATE RECIPROCAL, FLOATING POINT CONVERSION, ETC.

TWO ARGUMENT ADDER INSTRUCTION: PERFORMS ONE OF EIGHT OPERATIONS ON THE CONTENTS OF TWO OPERAND REGISTERS TO FORM THEIR SUM, DIFFERENCE, MAXIMUM, MINIMUM ABSOLUTE, ETC.

MULTIPLY INSTRUCTION: PERFORMS ONE OF FOUR VARIATIONS OF THE PRODUCT OF TWO OPERANDS.

DATA TRANSFER INSTRUCTION: MOVES THE CONTENTS OF ONE OF SIX SOURCE REGISTERS TO ONE OF 19 DESTINATION REGISTERS.

JUMP AND CALL INSTRUCTIONS: SPECIFY A (CONDITIONAL) JUMP WITHIN THE APU PROGRAM.

CONTROL OPERATIONS: INCLUDE RETURN JUMPS FROM SUBROUTINES AND INSTRUCTIONS TO CHANGE THE STATUS OF A SYSTEM FLAG.

Fig 10 Main memory addressing. Unit B control signal processing unit loads, from main memory, arithmetic addresser section (APS). Seven of 32 main memory bits designate memory location in APS, while remaining 25 bits are instruction bits

address specifies the location in arithmetic addresser memory where the instruction is to be loaded. The CSPU program resides in main memory. This program executes full-word (32-bit) and half-word (16-bit) instructions.

Host interface module (HIM) is the processor through which the unit B array processor and the host computer communicate. This module transfers blocks of data between the three memories and the host computer memory, converts data formats between the internal array processor format to or from that used by the host computer, and allows control information necessary to synchronize operations between the host computer and unit B. The HIM processor has a 32-word, 18-bit/word, programmable memory that is loaded by the CSPU.

The I/O processors are subprocessors that handle block transfers between external devices and the three memories in a programmed manner established by the CSPU. These I/O processors allow data transfers to occur without intervention of the host computer but under independently programmed control. A block transfer rate of 8 MHz is possible, while a programmed I/O (PIO) maximum rate of 1.8 MHz is feasible. All transfers are 8, 16, or 32 bits; parallel operation; and data may be packed or unpacked and multiplexed or demultiplexed on the fly in the PIO mode. Conversion between integer and floating point formats is not possible in the I/O processor.

In summary, the architectural design of unit B uses an asynchronous parallel structure to speed up arithmetic operations. The parallel asynchronous microprocessor design increases speed by complex interprocessor control. This asynchronous approach eliminates predictability of data flow and state timing such that a complex protocol is required between processor elements. Programs are difficult to write and to debug. Complex logical decisions based upon data values are also difficult, although complex I/O multiplexing is handled efficiently. Program and data memories are used efficiently, which should reduce overall cost.

Conclusions

A simple deduction regarding the overall superiority of either of the described array processors is not practical. Both attempt to achieve the goal of being a high speed array processor, but each approaches this goal in a drastically different manner (see "Comparison of Major Array Processor Performance Characteristics").

A few descriptors are common between both array processors. Both require a host minicomputer to initially load programs and to perform any necessary setup

Comparison of Major Array Processor Performance Characteristics

	FPS AP-120B	CSP MAP 300
Ideal Floating Point Operation, Maximum Rate	12.0M floating point calculations/s	14.25M floating point calculations/s
Maximum Block I/O Rate (DMA)	3M words/s	8M words/s
Maximum Programmed I/O Rate	Not Available	1.8M words/s
Capability to Perform PIO Data Manipulation	No	Yes
Internal Floating Point Word Size	38 bits (27-bit mantissa)	32 bits (24-bit mantissa)
Floating Point Mantissa Guaranteed Precision	21 bits	21 bits
Floating Point Format Dynamic Range	$10^{\pm 308}$	$10^{\pm 19}$
Number of Floating Point Accumulators	64	32
Fast Access Table Memory	Up to 64k words	Not Available

control of peripheral devices and overall sequencing of processing events. Both achieve significant numerical processing speed improvements over a typical minicomputer while performing repetitive numerical operations. Neither performs efficiently while performing complex logical operations based on data values. Beyond these general statements, the distinctions become more apparent.

Unit A (AP-120B) has a greater numerical dynamic range and carries greater numerical precision in its calculations. An overall reduced error from repetitive calculations upon data is a definite advantage that unit A holds over unit B (MAP-300).

The synchronous parallel operation of unit A versus the asynchronous parallel operation of unit B manifests itself in relative complexity of conceptualization and sequencing control. Unit A, although more difficult to program than a typical minicomputer, should be vastly easier to program than unit B. This simplicity is obtained by inefficient usage of the high speed (but high cost) program control memory in unit A.

On the other hand, unit B provides more flexibility in I/O control and more efficient utilization of both program and data memory. A definite disadvantage of unit B is the small program memories in all but the CSPU. This limitation necessitates that the CSPU reload the submicroprocessor control programs whenever a functional processing change is made, requiring suspension of numerical processing for the many microseconds needed to load the particular program memory with the necessary software code.

The 64 accumulators and the fast access table memory of unit A provide a potential processing advantage over the 32 accumulators and no table memory in unit B. Alternately, the parallel adders and multipliers of unit B provide a comparable arithmetic speed-up advantage over the pipeline adder and multiplier of unit A.

Finally, selection between unit A and unit B should be based primarily upon programming cost, I/O requirements, minimum acceptable memory configuration, and final cost for each application configuration. Although difficult and time consuming, benchmark programs of specific processing algorithms, which are typical of the machine's intended application, would best illustrate the advantage of one array processor over the other.

Bibliography

R. A. Caspe, "Array Processors," *Mini-Micro Systems*, July 1978, pp 54-63

CSP, Inc, "Macro Arithmetic Processor Systems (MAP-100/200/300) Programmers Reference Manual," Doc No. JB 6000-001-00, CSP Inc, 209 Middlesex Tpk, Burlington, MA 01803, May 1975

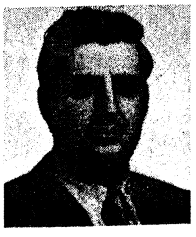
Floating Point Systems, Inc, "AP-120B Array Transform Processor Handbook," Doc No. 725g-01, Floating Point Systems, Inc, Portland, Ore, 1976

A. S. Margulies, "Array Processing Eases High Speed Vector Computation," *Digital Design*, June 1978, pp 52-56

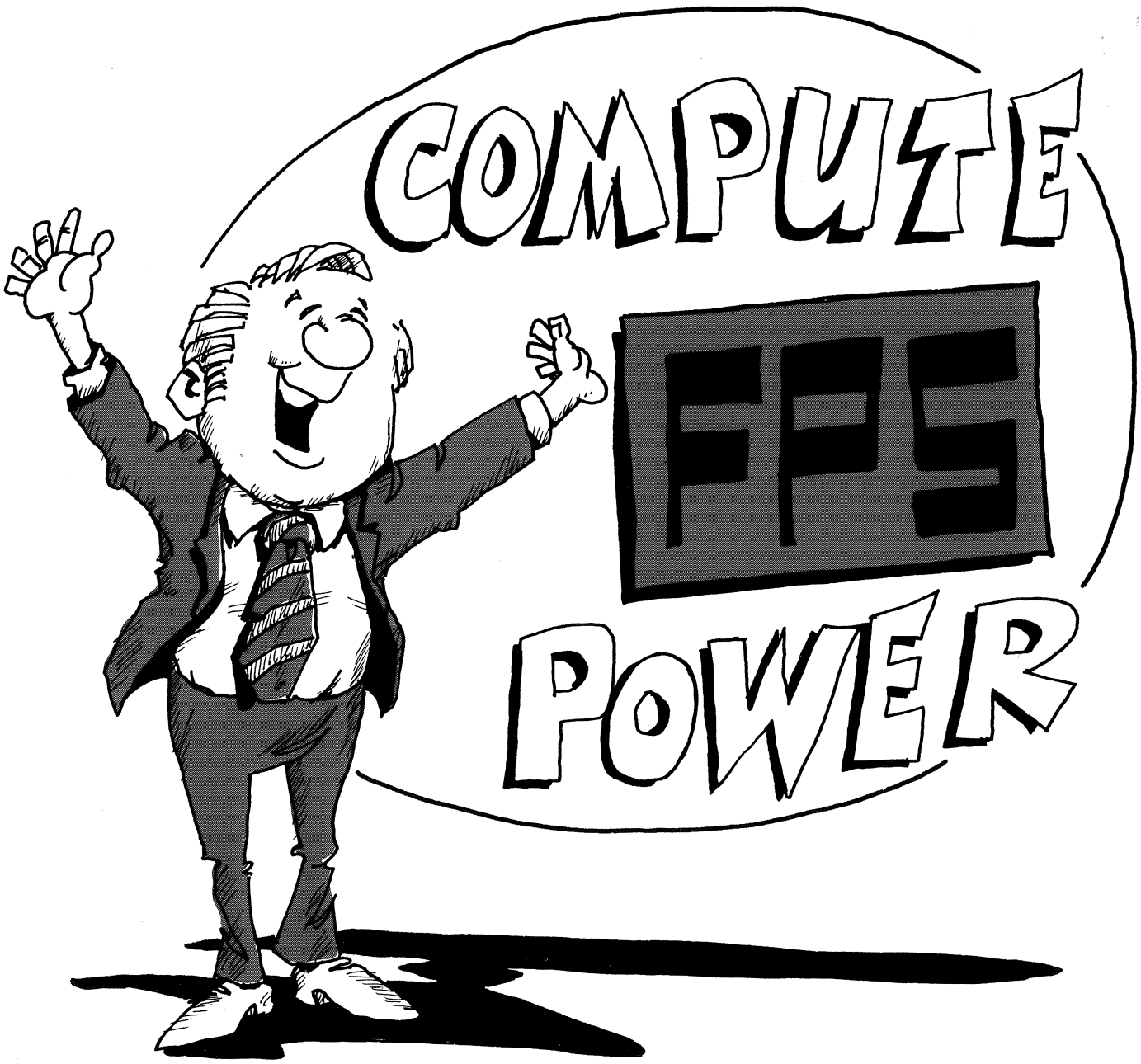
J. Meyn, "Processing Digitized Video In Realtime," *Computer Design*, Dec 1978, p 104

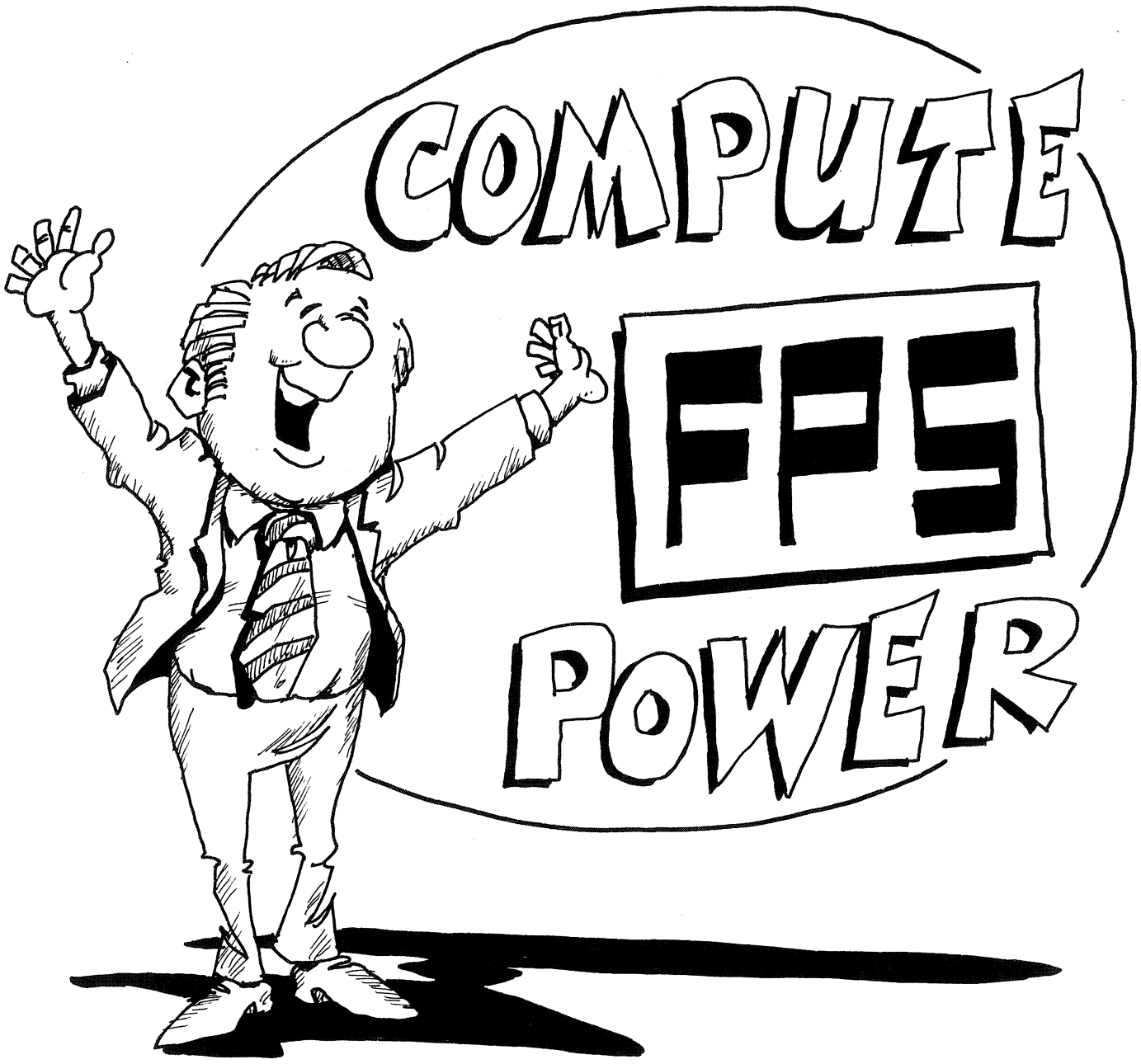
C. N. Winningstad, "Scientific Computing On a Budget," *Datamation*, Oct 1978, pp 159-164

W. R. Wittmayer, "Array Processor Provides High Throughput Rates," *Computer Design*, Mar 1978, pp 93-100



Stephen Hufnagel is a research scientist at Applied Research Laboratories, The University of Texas at Austin, where he is involved in an exploratory development project entailing application of minicomputers to monitor, process, and control acoustic systems in a realtime environment. He holds a BS degree in mathematics from The University of Texas and has done graduate studies in electrical engineering and computer science there.

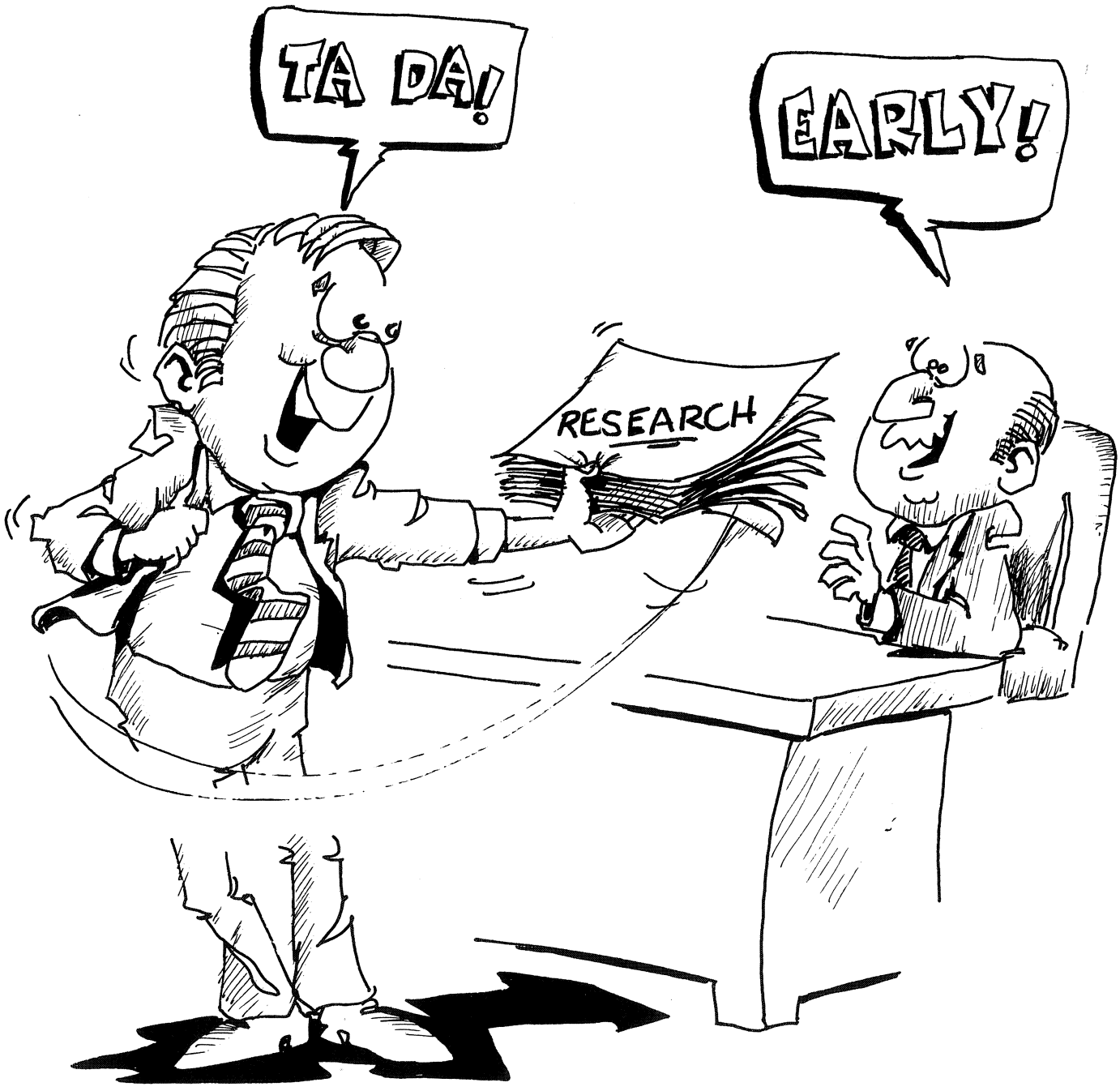

COMPUTE
FPS
POWER



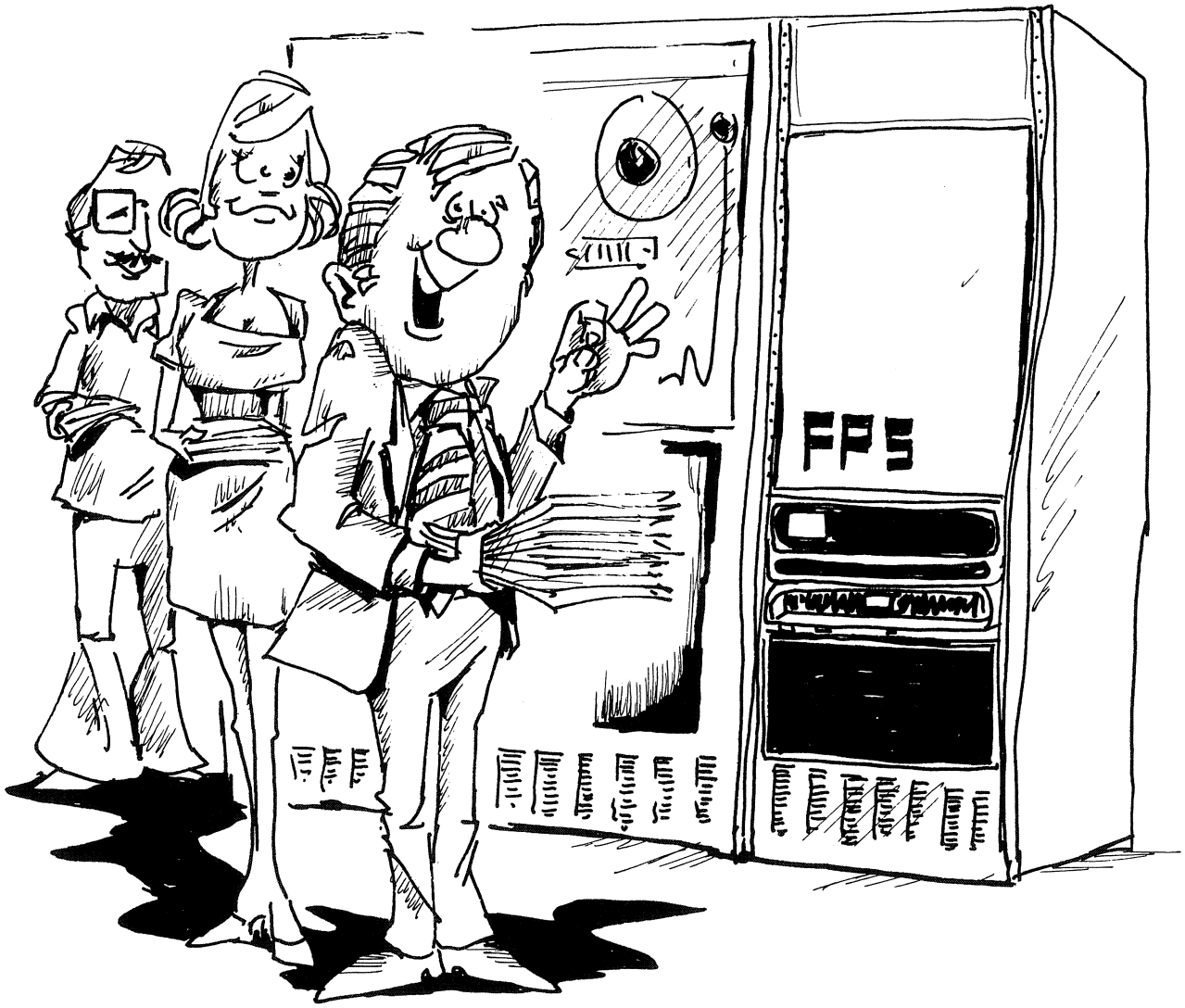
Welcome to the National Computer Conference and to Floating Point Systems. We would like to show you the inexpensive way of getting extraordinary compute power . . . the FPS Array Processor.

Floating Point Systems has over 500 installations worldwide using its scientific computers, and compute power means something special to each of our customers.

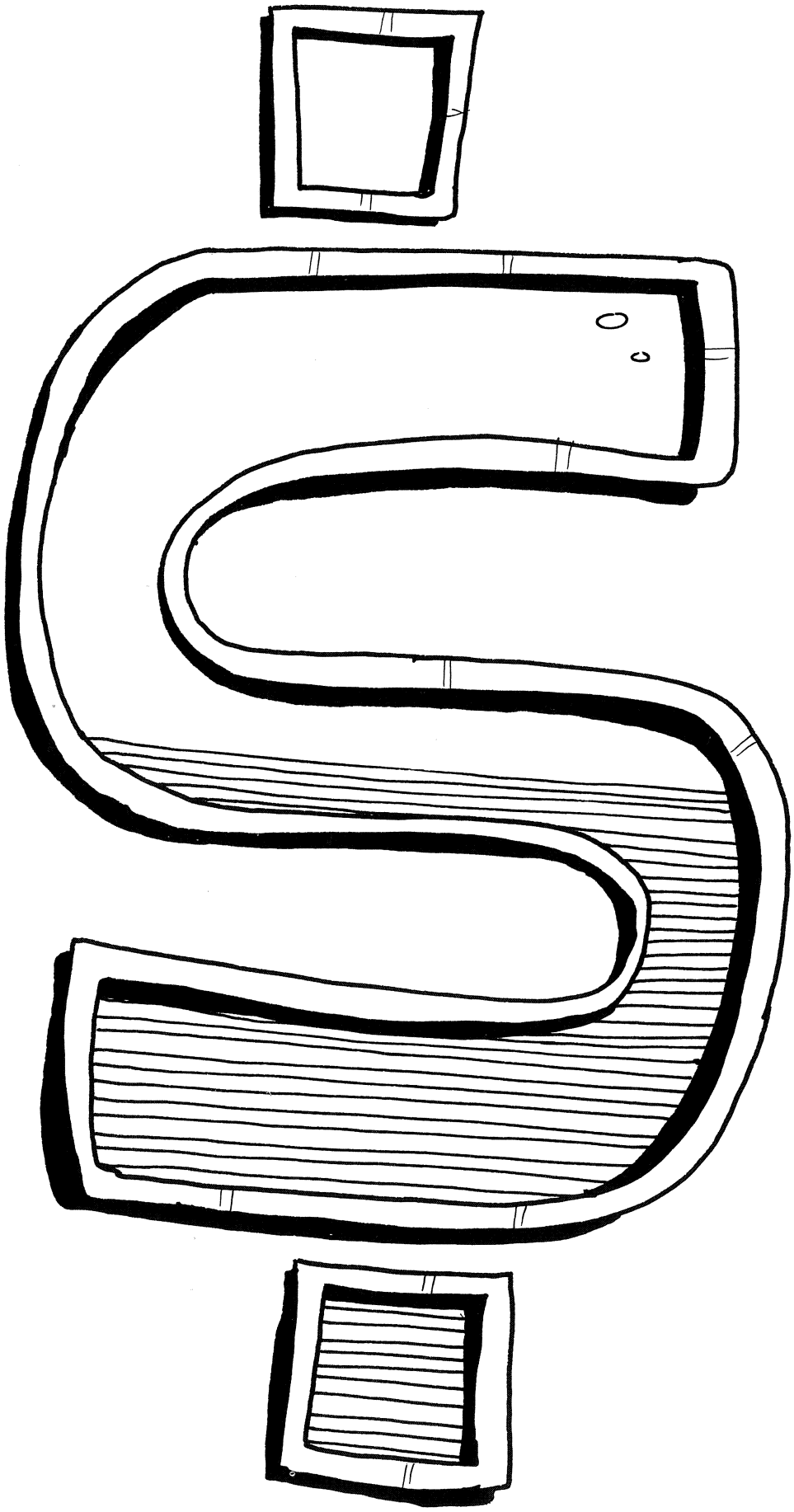
To some, it means being first in their industry with a powerful new product.



To others, compute power means getting results from research months ahead of schedule.



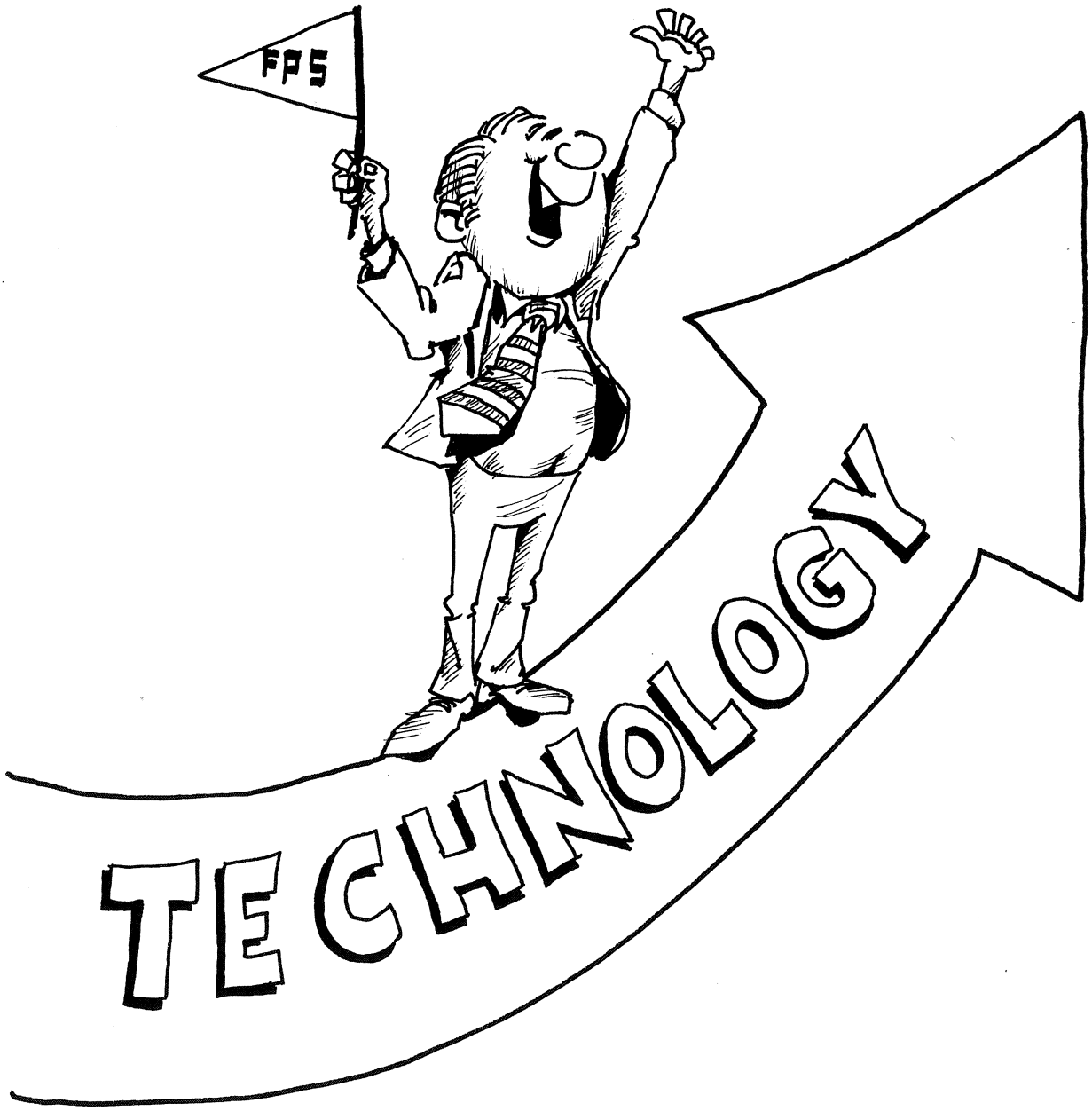
And to others it means more people can be accommodated at their computer center faster.



To all of our customers compute power means money. Money saved in processing time, in product development time, and money saved by having a hundred-thousand dollar system doing the work of a multi-million dollar system.

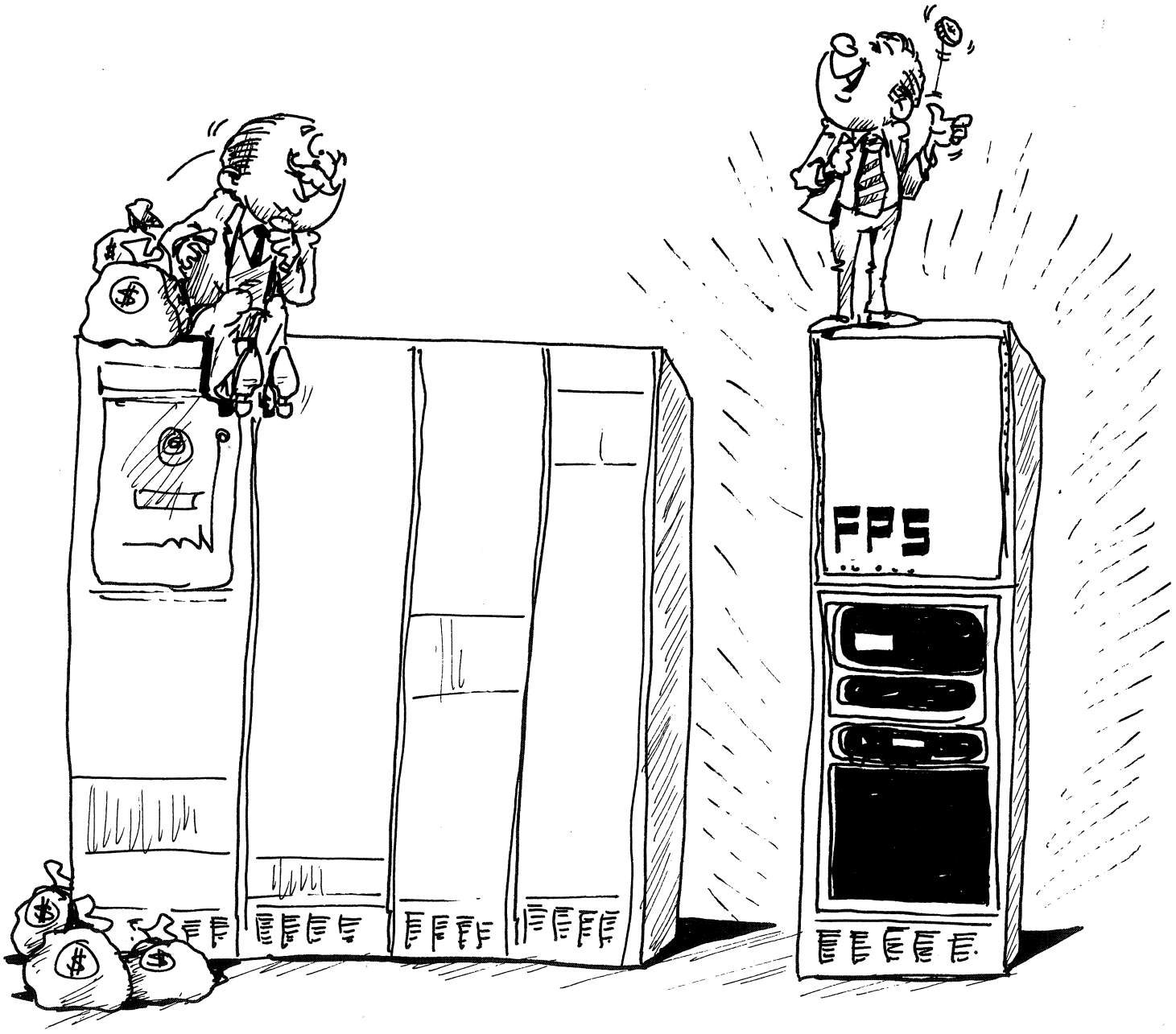
- RESEARCH
- PROCESS CONTROL
- PRODUCT DEVELOPMENT

It doesn't matter whether it's research, process control, or product development, they all demand compute power available from Floating Point Systems.



The exponential growth of technology has created a dramatic need for economical compute power. It is not surprising that our customers demand compute power for their business.

Of course, heavy compute power has always been available but at a tremendous cost. It usually meant buying a large computer system which would cost millions of dollars.



Floating Point Systems has changed all that! We have designed a supercomputer at minicomputer prices! It gives your computer the computational power you need but at a fraction of what you would pay for a large system. By fraction, we mean a $50,000 array processor can be added to your minicomputer to provide you with the computational power of a CDC 7600. If you have a mainframe, we can add the same power to your system.

- COMPLEX SCIENTIFIC CALCULATIONS
- DATA REDUCTION
- SIMULATION AND MODELING
- IMAGE PROCESSING

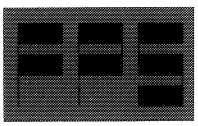
Ask yourself what processing does your organization need that has been out of reach because of cost. Whether your application is--

Complex scientific calculations
Data reduction
Simulation and modeling
or
Image processing,

you can afford to consider it now.

Come inside and let us help you add compute power to your system.


FPS
FLOATING POINT
SYSTEMS, INC.
FPS
FLOATING POINT
SYSTEMS, INC.



FLOATING POINT SYSTEMS, INC.

CALL TOLL FREE 800-547-1445

P.O. Box 23489, Portland, OR 97223 (503) 641-3151,
TLX: 360470 FLOATPOINT PTL
In Europe & UK: Floating Point Systems, SA Ltd.
7 Rue du Marché, 1204 Geneve, Switzerland 022-280453,
TLX: (845) 28870 FPSE CH

UNITED STATES AND CANADIAN SALES OFFICES

• Riverdale, Maryland	301/699-9400	• Houston, Texas	713/977-7769
• Red Bank, New Jersey	201/747-6810	• Detroit, Michigan	313/553-9890
• Fairfield, Connecticut	203/259/8351	• Los Angeles, California	714/768-0555
• Philadelphia, Pennsylvania	215/688-8310	• Palo Alto, California	415/321-6051
• Orlando, Florida	305/647-2501	• Phoenix, Arizona	602/982-3385
• Huntsville, Alabama	205/883-5530	• Portland, Oregon	503/620-9350
• Boston, Massachusetts	617/848-6314	• Denver, Colorado	303/647-8319
• Dallas, Texas	214/783-1935	• Ottawa, Ontario	613/820-9608
• Chicago, Illinois	312/991-7177		

INTERNATIONAL SALES OFFICES

• Geneve, Switzerland	022-280453
• Reading, Berkshire, England	0734-50181
• Munich, Germany	089-619008-09
• Paris, France	687 62 31