

JANUARY 1973

# HEWLETT-PACKARD JOURNAL



# An Economical Full-Scale Multipurpose Computer System

*This is the first 16-bit computer system to have a hardware stack architecture and virtual memory. It handles time-sharing, batch processing, and real-time operations in several languages concurrently.*

by Bert E. Forbes and Michael D. Green

**T**HE HP 3000 COMPUTER SYSTEM is Hewlett-Packard's first full-scale multipurpose computer system. Its primary objective is to provide, at low cost, a general-purpose\* computer system capable of concurrent batch processing, on-line terminal processing, and real-time processing, all with the same software. Many users can access the system simultaneously using any of several programming languages and applications library programs.

The HP 3000 (Fig. 1) is an integrated software-hardware design. It was developed by engineers and programmers to provide a small computer capable of multiprogramming. Unlike many computers of the past, it was not built by the engineers and turned over to the programmers to see what they could do with it.

Helping define the objectives for the HP 3000 was HP's long experience with both customer and internal use of 2100-Series Computers and 2000-Series Time-Shared Systems. These computers and systems have been widely used in educational, instrumentation, industrial, and commercial applications. These are also expected to be the primary applications areas for the HP 3000 (see page 7).

A comprehensive set of software and the hardware to support it has been developed for the HP 3000. Software includes the Multiprogramming Executive operating system, several programming language translators, and an applications library.

## Architectural Features

The scope of the software for the HP 3000 requires certain capabilities in the computer on which the software is to run. Among these are efficient program segmentation, relocation, reentrancy, code

\*"General-purpose" means that a user is not restricted to a single application, but can readily write programs to fit his own application, whatever it might be.

sharing, recursion, user protection, code compression, efficient execution, and dynamic storage allocation. All are provided in the HP 3000 design.

**Efficient program segmentation** makes it possible to run programs which are much larger than the available memory without incurring a large overhead. Much of the power and flexibility of the HP 3000 comes from the virtual memory that results



**Cover:** *Although it could be mistaken for an organist in concert, this photo actually shows the new HP 3000 Computer System harmoniously coordinating the activities of several users—it handles multiple processing modes, users, and languages all at the same time. The panels with the red lights in the foreground are control panels used for hardware maintenance and system checkout.*

## In this issue:

- An Economical Full-Scale Multipurpose Computer System*, by Bert E. Forbes and Michael D. Green . . . **page 2**
- Central Bus Links Modular HP 3000 Hardware*, by Jamshid Basiji and Arndt B. Bergh . . . **page 9**
- Software for a Multilingual Computer*, by William E. Foster . . . **page 15**
- Single Operating System Serves All HP 3000 Users*, by Thomas A. Blease and Alan Hewer . . . **page 20**



**Fig. 1.** HP 3000 Computer System has multilingual and multiprogramming capabilities usually found on much larger systems.

from the segmentation capabilities of the system.

Swapping of programs and data is made easier by an automatic **relocation** technique that is part of the addressing structure of this multiprogramming computer. The operating system doesn't have to take time to adjust all addresses in a program, nor does it have to put something in the same physical location every time.

**Reentrancy** is a property of HP 3000 code. It makes it possible for a given sequence of instructions to be used by several processes without having to be concerned about the code being changed or temporary variables being destroyed by the other processes.

Automatic relocation and reentrancy make **code sharing** possible. It would be extremely wasteful of main memory to keep multiple copies of programs in memory. In the HP 3000, one copy of a program can be shared by many processes.

Another consequence of reentrancy is **recursion**, or the ability to have a routine call itself. The hardware stack architecture of the HP 3000 plays an important role in recursive calls.

One of the key items in designing a multiprogramming operating system is that of **user isolation and system protection**. If the operating system and the users are not completely protected from the intentional or unintentional destructive actions of another user, the system will crash so often as to

be unusable. HP 3000 protection covers programs, data, and files that exist in the system.

In a small-word-size machine, the amount of addressable memory is limited. To take full advantage of it, the HP 3000 has **dynamic storage allocation**. All temporary and local variables are assigned physical memory only when needed at procedure or block entry and are deallocated upon exit.

The HP 3000's unified real-time, terminal-oriented, and batch environment is accomplished without the use of fixed or variable memory partitions. Instead, priorities are used to control system resources. Partitioning, it was felt, places arbitrary restrictions on memory, the most valuable resource in a multiprogramming system.

#### **Multiprogramming Executive Operating System**

The HP 3000 has a single operating system called the Multiprogramming Executive (MPE). MPE is a general-purpose system that can handle three modes of operation concurrently. In time-sharing, one or more users can interactively communicate with the system via computer terminals. In batch processing, users can submit entire jobs to be performed by the system with no interaction between the system and the user. In real-time operations, tasks are dependent upon the occurrence of external events and must be performed within critical time periods.

MPE also provides many services to users, such

as input/output handling, file management, memory management, and system resource allocation and scheduling.

There are many advantages to an operating system of this sort. For example, subsystems (compilers, applications programs, etc.) need not be customized for different operating systems or configurations. In fact, the same subsystem can be used concurrently by an interactive user and by a batch user. Another advantage is that software can be generated much more efficiently because the operating system already performs many of the more difficult tasks. Also, with a single operating system, it is easier to attain consistency throughout the various software subsystems, thereby simplifying the user/system interface.

Although there is only a single operating system on the HP 3000, it can be adapted to operate in a number of different hardware configurations, each tailored to the needs of its users (Fig. 2). Thus, one installation may run only small batch processing jobs using a card reader and a line printer, while another may add a number of time-sharing terminals. The same software is used by all these installations.

### **Programming Languages**

Several different programming languages have been developed for the HP 3000. Most important of these is SPL, the Systems Programming Language. This is a higher-level programming language designed specifically for systems programming. Almost all of the HP 3000 software has been developed using SPL, the few exceptions being some of the applications programs.

The reasons for using a higher-level language rather than an assembly language for systems programming are much the same as those for using a higher-level language for applications programming. It's possible to write and debug programs more quickly, to modify them more easily, and to make them more reliable and easier to read and understand. Furthermore, programs often perform better because more time can be spent on general methods than on coding details. Improving programs by rewriting substantial sections of code is not distasteful, as it often is when programs are written in assembly language. In general, in a given period of time much more software can be developed by using SPL than by using a lower-level assembly language.

Since SPL is designed for HP 3000 systems programming, it was necessary to give SPL programmers easy access to all the features of the central processor. For this reason, much of the syntax is

based on these features, and the machine code generated by the compiler is related in an obvious way to the higher-level statements in the language.

Other programming languages which have been developed for the HP 3000 are FORTRAN, COBOL, and BASIC. SPL, BASIC, and FORTRAN are all recursive, that is, programs, procedures, and subroutines can call themselves. HP 3000 software also includes scientific and statistical applications program libraries, and text editing and formatting facilities.

### **Program Environment**

Traditionally, 16-bit computers have been von Neumann-like machines with little or no distinction between program code and data. In a multi-programming environment there is much to be gained from separating the two. In the HP 3000, a typical user's environment consists of one or more program code segments and a data segment (Fig. 3). All code is nonmodifiable while active in the system. Overlay techniques can therefore be used (that is, new code can be written over old code) without having to write the old code back out on the swapping disc, since an exact copy already exists there. The data area consists of global data (data common to several procedures) and a push-down stack that is handled automatically by the hardware.

### **Code Segmentation**

In the HP 3000, code is grouped into logical entities called segments, each consisting of one or more procedures. Each segment may be up to 16K words long. Programs are normally broken into segments by the user, although he may choose not to do this and his program will run as a single segment unless it is too large, in which case an error message will be generated.

There is a master directory, the Code Segment Table (CST), that contains one entry for each segment that is currently active on the system. The CST is maintained by the operating system and is used by the central processor for procedure entry and exit. The table doesn't occupy a fixed position in memory but its address is always stored in absolute location 0.

Each two-word CST entry contains the beginning address and the length of the code segment. There are also four bits that are used by the central processor. One of these, the reference bit, is used to implement a software least-recently-used overlay algorithm. Another, the trace bit, causes a procedure call to the trace routine if set. The mode bit specifies whether the segment will be run in privileged or user mode. The absent-from-main-memory

## HP 3000 COMPUTER SYSTEM

### 30000A MAINFRAME

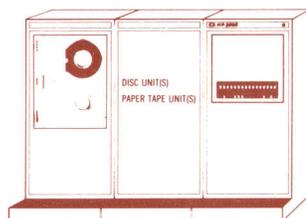
INCLUDES: CPU, cabinets, power supplies, card cages, multiplexer channel, 64K bytes core, memory controller, system control desk, console/terminal interface, internal system clock.

- **AC Power Options for total system:**  
Standard—120/208V, 3 phase, 60 Hz  
-015—230V, Single Phase, 50 Hz  
-025—120/240V, Split Phase, 60 Hz

- **Color Options—System Accent**  
Standard—Sun Gold  
-050 Woodgrain  
-051 Marine Blue  
-052 Red

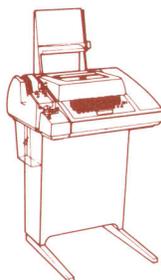
- **Additional Memory Options—total system capability in bytes.**  
-101 64K, 2 mcu—no interleaving  
-120 80K, 2 mcu—no interleaving  
-140 96K, 2 mcu—no interleaving  
-160 112K, 2 mcu—no interleaving  
-180 128K, 2 mcu—no interleaving  
-181 128K, 2 mcu—2 way interleaving  
-182 128K, 4 mcu—no interleaving  
-183 128K, 4 mcu—4 way interleaving

(mcu = module control unit)



Required Items  
1-Magnetic Tape Drive 1-Console  
1-Disc

### CONSOLES AND TERMINALS



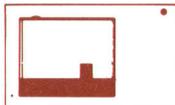
• 30123A CRT Console/Terminal

• 30124A ASR-33—Console/Terminal

### PAPER TAPE



Reader (Racked)  
• 30104A Reader and interface.



Punch (Racked)  
• 30105A Punch and interface.

### CARD PUNCHES



• 30112A Punch;  
250 cpm

### TIME SHARING, PERFORMANCE, RTE, AND OTHER OPTIONAL EQUIPMENT

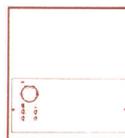
- 30032A Asynchronous Multiplexer 16 terminals—hardwired  
-001 103 Data Set  
-002 103 and 202 Data Sets (order either 001 or 002, not both)
- 30055A Synchronous Single Line Controller, 201 or 208 Data Sets, 9600 bps with cable
- 30030A First High Speed Channel

- 001 Second High Speed Channel
- 30390A Expansion Bay (Above normal requirements)
- 30050A Universal Interface (UI); Ground Level True, TTL  
-001 Universal interface (UI) Positive Level True, TTL
- 30051A Universal Interface (UI) Differential Levels
- 30031A Second Clock

### DISCS



Cartridge (Racked)  
• 30110A 4.9MByte Cartridge Disc and 4 drive interface.  
-010 Additional drive.

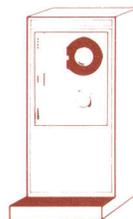


Fixed Head (Racked)  
• 30103A 1MByte and interface.  
-001 2MByte total.  
-002 4MByte total.



11 High Disc - Removable  
• 30102A 47MByte drive and 8 Drive Interface.  
-010 Additional Drives.

### MAGNETIC TAPE UNITS



- 9-Track (Racked)  
• 30115A 800 cpi; 45 ips drive and 4 drive interface.  
-100 1600 cpi; 45 ips. Master drive and 4 drive interface.  
-200 800 cpi; 45 ips drive. Additional Unit.  
-300 1600 cpi; 45 ips. Master drive. Additional Unit.

-400 1600 cpi; 45 ips. Slave<sup>®</sup> drive. Additional Unit.  
® (Must be used with 1600 cpi master on same interface)

#### 7-Track (Racked)

- 30117A 200/556/800 cpi, 45 ips drive and 4 drive interface.  
-010 200/556/800 bpi, 45 ips drive. Additional Unit.

### CARD READERS



- 30106A 600 cpm
- 30107A 1200 cpm  
-001 Dual Read Station

### LINE PRINTERS

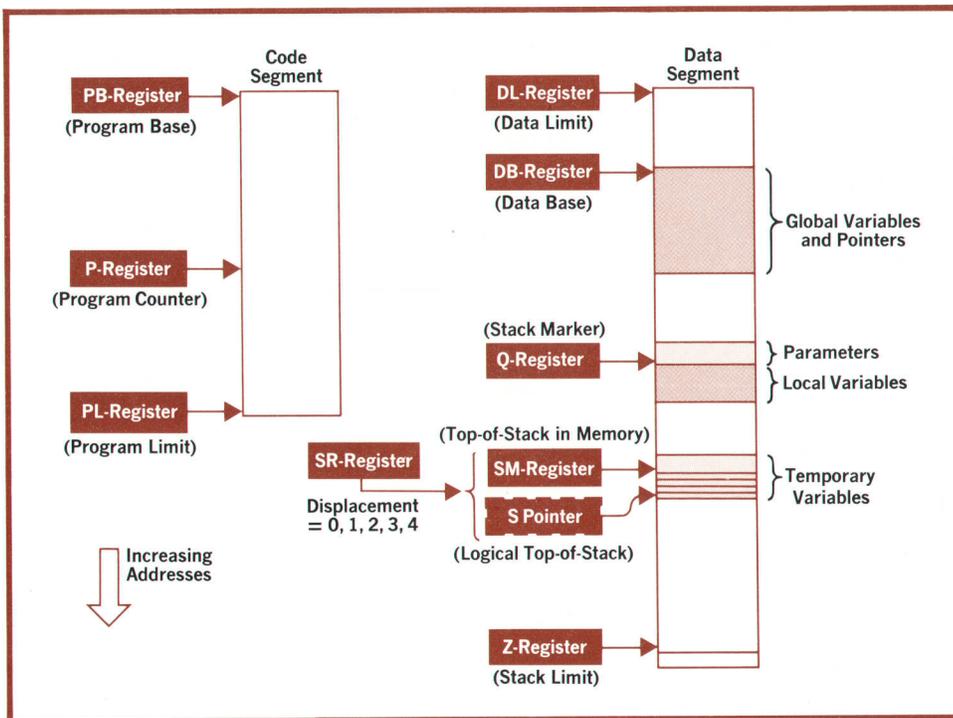


- 30108A 200 lpm; 64 Char.  
-001 130 lpm, 96 Char.
- 30109A 600 lpm; 64 Char.  
-001 400 lpm; 96 Char.

### SOFTWARE PRODUCTS

- 32000A MPE/3000 (includes Systems Diagnostic Monitor /30000, Compiler Library, Utilities)
- 32100A SPL/3000
- ▲ 32101A BASIC/3000
- ▲ 32102A FORTRAN/3000
- ▲ 32201A EDIT/3000
- ▲ 32202A FORMATTER/3000
- ▲ 32204A STAR/3000
- ▲ 32205A SCIENTIFIC LIBRARY
- Standard Software
- ▲ Optional Software

Fig. 2. HP 3000 Computer Systems are modular and can be configured to fit a variety of applications. The same software is used by all configurations.



**Fig. 3.** Code and data are kept separate in the 3000 Computer System. Code is never modified and can be shared by several users. Code segmentation gives the system virtual memory capability. Data segments are organized as pushdown stacks.

bit causes a procedure call to the make-present routine if set, and it implies that the second word of the CST descriptor is a disc address. The maximum number of entries in the CST is 255.

Every procedure call must go through the Code Segment Table and must check the absent-from-main-memory bit. This is part of the virtual memory implementation of the HP 3000. If the procedure called is in a program segment that isn't in main memory, the required segment is automatically brought in. When a segment is given control of the central processor, the program base (PB) and program limit (PL) registers are set from the CST entry of that segment.

While code segmentation is normally specified by the user, data segmentation is handled by the MPE/3000 operating system. A normal user has only one data segment, which is limited to 32K words. Additional data segments may be requested.

### Relocatable Code

Relocation is the normal mode of operation in the HP 3000 because of its relative addressing capability. All addressing is relative to hardware registers.

Fig. 4 shows the memory reference instruction format. The address mode bits have been Huffman-coded to give the maximum displacement range on the most frequently used codes.

In the code segment, normal addressing is relative to the program counter register (P). Indirect

addressing is similar except that the content of the indirect cell is assumed to be relative to its own location.

In the data segment, the addressing modes are designed to match the types of data encountered in a procedure-oriented language. Fig. 3 shows the organization and common use of the data area. Global variables and pointers are stored relative to the data base (DB) register. The DB+ mode has a direct range of up to 255 words without indexing or a 64K word range with indexing.

### Stack Operation

The stack concept,\* which on the HP 3000 is fully used for the first time in a 16-bit machine, allows dynamic storage allocation on a procedure level. The stack is the area of a user's data segment between the DB register and the stack pointer (S).

Local stack storage in a procedure is allocated only upon entry and is automatically freed upon exit. This allows reuse of that area of memory by other parts of the program. The stack also provides automatic temporary storage of intermediate results until they are needed later in a computation. This is transparent to the programmer, and the compiler doesn't have to be concerned with saving and restoring registers.

Parameters that are passed to procedures are

\*A stack is a linear collection of data elements which is normally accessed from one end on a last-in-first-out basis. An everyday example is a stack of plates in a plate warmer in a cafeteria.

## A Computer for All Reasons

Education and Instrumentation are traditional fields for HP, and the HP 3000 Computer System significantly enhances the company's capabilities in these areas. The new system is also well suited to advanced industrial and commercial applications.

### Education

HP computers entered the education field in 1968. The HP 2000A Time-Shared BASIC System, along with its successors, provided cost-effective computer aided instruction (CAI), problem-solving, and computer science education. A math drill and practice program, an instructional dialog facility, and an instructional management facility are available to the teacher for use on the HP 2000. These programs are written in HP BASIC and are therefore upward compatible with the HP 3000. In addition to these programs, there are other CAI packages available for use on both HP systems.

In addition to the time-shared CAI use of the HP 3000, the Multiprogramming Executive operating system allows simultaneous batch mode computation. This permits a school to use the computer for administrative tasks concurrently with CAI, giving a more cost-effective solution to the needs of school systems. Many secondary schools will also be able to teach programming and other computer science concepts using the multiprogramming capability of the HP 3000.

Junior colleges and small four-year colleges, to keep costs down, often find it necessary to have only one computer for all their activities. The HP 3000, with its simultaneous multilingual time-sharing and batch operating modes, has the ability to tackle diverse computing needs. In addition to these two modes, real-time experiments may also be handled by the operating system, and this makes the system useful to university scientific departments.

Because the HP 3000 was designed around the latest concepts in computer science, it has many features in hardware and software that university professors have been teaching in recent years. The 3000 should provide a computer science department with a machine that can be used not only as case study of advanced architecture, but also as a vehicle for operating system study. The modular structure of the software allows students to rewrite small portions of the system as projects and then try them in the operating system. It would be too large a task to write a whole system in a semester, but a small self-contained

module is the right size for a term project.

### Instrumentation

While the education field is mainly interested in the time-sharing and batch modes of operation under the Multiprogramming Executive (MPE), the instrumentation field makes extensive use of the compatible real-time capability. Previous systems generally had one or the other, but not all three in a unified environment.

MPE provides the ability to collect data and control processes in real time while allowing the data so generated to be accessed through the common file system by terminal-oriented and batch mode programs. This multi-mode capability is a natural extension and combination of the real-time executive and time-sharing systems that use the HP 2100 family of computers.

### Industrial/Commercial

The HP 3000 will find many industrial and commercial applications. One reason is that it is designed to support hierarchical computing systems. The data-base handling capabilities of MPE, along with a powerful and wide-bandwidth I/O structure, make the 3000 a good middleman computer. It will have extensive data communication facilities for connection to a large general-purpose computer and will be able to control several minicomputers on the other end of the hierarchy.

There will be many instances of this computer-to-computer connection in the future. Standard software protocols and hardware interfaces are being developed for the HP 3000 to support these systems. The HP 2100 family provides compatible minicomputer facilities in systems where the 3000 is the host computer. Intercomputer links may be by direct connection or by modems over common carrier facilities.

Commercially oriented languages and data base management systems currently in development will give HP the ability to develop and support commercial applications such as on-line inventory management, order entry and production control. The hierarchical computing capability combined with this business data-processing software will make the HP 3000 more and more useful in industry and commerce, particularly if the strong trend toward distributed processing continues as expected.

pushed onto the stack before the procedure call. When the procedure call occurs the status of the presently executing code segment is stored on the stack and the Q register is set to point at the top of the stack (S). Parameters are then accessed by Q- addressing, while the local variables used by the procedure are accessed by Q+ addressing, as shown in Fig. 3.

Upon exiting from a procedure the operating system retrieves the status of the previously executing code segment from the stack and returns control to the instruction following the procedure call.

Addressing in the negative direction with re-

spect to the stack pointer (S) register is useful for accessing temporary results left on the stack during processing. The area between the data limit (DL) register and DB may be addressed only indirectly and is used for such purposes as storing symbol tables and the like.

### Reentrant Code

The separation of code and data, the use of a pushdown stack with Q+ and Q- addressing modes and the nonmodification of code make reentrant code the natural way to write HP 3000 programs. Reentrant code, in conjunction with the

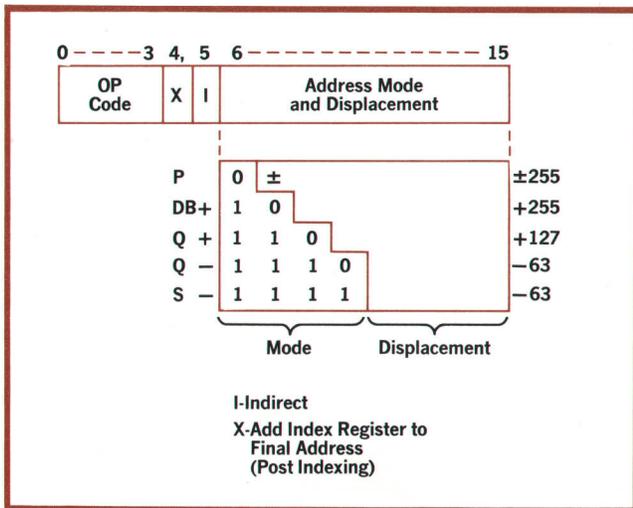


Fig. 4. Memory reference instruction format. All addressing is relative to hardware registers, making code and data easily relocatable.

use of the Code Segment Table as the master directory of all active segments, allows code segments to be shared between users. Control is transferred through the CST to the proper segment number of the shared code as determined by the loader when the segment was made active. Thus only one copy of a compiler or a library or the operating system intrinsics need be available, saving valuable space in main memory.

#### Protection Features

User isolation and protection takes several forms on the HP 3000. Programs may execute in one of two modes: privileged or user. In privileged mode no bounds checking is done except for stack overflow ( $S > Z$ ), and all instructions are available for use. All system interrupts including external (I/O) interrupts are handled on a separate interrupt control stack so the user running when the interrupt occurs is fully protected. In user mode, access is limited to within the user's own code and data areas.

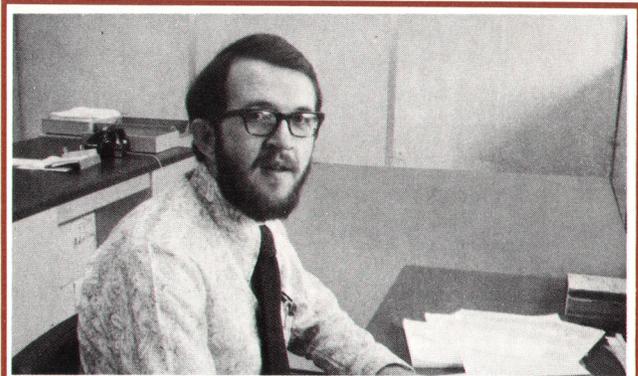
In addition to the hardware memory protection, files are protected by the MPE/3000 file management system. Access to files may be controlled at several levels which range from unrestricted access by anyone to controlled access available only to the creator of the file.

#### Modular Hardware Organization

HP 3000 hardware is organized on a modular basis. A major feature is the central data bus, which can service up to seven independent and asynchronous modules. These can be central processors, memory modules, and/or various types of input/output channels including a high-speed selector

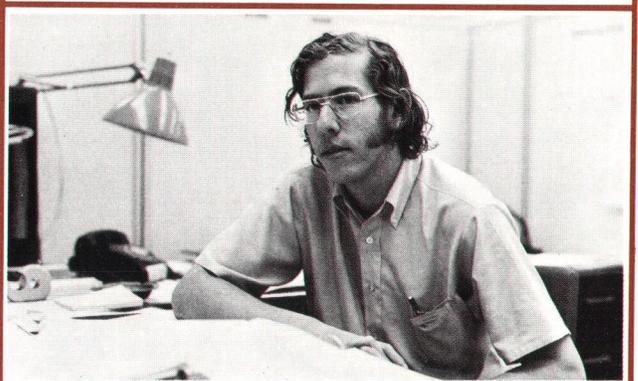
channel capable of transferring data at a rate of 2.8 megabytes per second.

Modules attached to the bus are technology-independent. Thus the memories may be magnetic core, semiconductor, or anything else. Up to four memory modules can be attached to the bus, and these can be interleaved (two-way or four-way).



#### Bert E. Forbes

Bert Forbes has been designing computers for HP since 1967. He was project manager for the HP 3000 CPU and has several patents pending as a result of that project. He's a member of ACM and the author of articles on computer architecture and integrated circuits for mini-computers. Now at HP's Geneva, Switzerland, data center, he's supporting the European introduction of the HP 3000. Bert received his B. S. degree in electrical engineering from Massachusetts Institute of Technology in 1966 and his M. S. E. E. degree from Stanford in 1967. He's also done work towards the Ph. D. degree. He's an amateur photographer and a connoisseur of fine wines, and is active in church youth and social-action groups.



#### Michael D. Green

Mike Green came to HP in 1966. He's been project manager for ALGOL/2116, 2000A Time-Shared BASIC, and BASIC/3000, and he's currently project manager for MPE/3000. Mike graduated from Columbia University in 1964 with a B.S. degree in mathematics, then got his M.S. degree in computer science at Stanford University in 1966. He's a member of ACM. For relaxing away from the world of computers, he favors bicycle touring and chess.

# Central Bus Links Modular HP 3000 Hardware

*Sharing the bus can be one or more CPU's, I/O processors, memory modules, high-speed I/O channels, and special devices. The microprogrammed CPU's have a procedure-oriented stack architecture.*

**by Jamshid Basiji and Arndt B. Bergh**

**O**N THE HARDWARE LEVEL, the HP 3000 Computer System consists of independently functioning modules communicating over a high-speed multiplexed central data bus (Fig. 1). The modules may include one or more central processing units (CPUs) and input/output (I/O) processors, one to four memory modules, one or more selector channels for high-speed input/output, and one or more special-purpose modules. Hardware modularity makes the system flexible and expandable, and leaves the door open for future performance improvements through new technologies such as faster memories.

The memory now available is a magnetic core memory that has a cycle time of 960 nanoseconds. Optional is an interleaved addressing capability that places sequential addresses in different memory modules. Memory modules can operate concurrently. With interleaving, the system can support a 5.7 megahertz byte data rate.

The 3000 CPU is a microprogram-controlled processor. It has a stack architecture and special hardware to make procedure execution very efficient. Instructions are implemented in microprogrammed read-only memories, making possible a powerful instruction set with some instructions resembling those of higher-level languages.

The data for each user is organized as a data stack. In general, a stack is a storage area in core memory where the last item stored in is always the first item taken out. The stack structure provides an efficient mechanism for parameter passing, dynamic allocation of temporary storage, efficient evaluation of arithmetic expressions, and recursive subroutine or procedure calls. In addition, it enables rapid context switching—21 microseconds to establish a new environment when an interrupt

occurs. In the HP 3000, all features of the stack (including checking for overflow and underflow) are implemented in hardware.

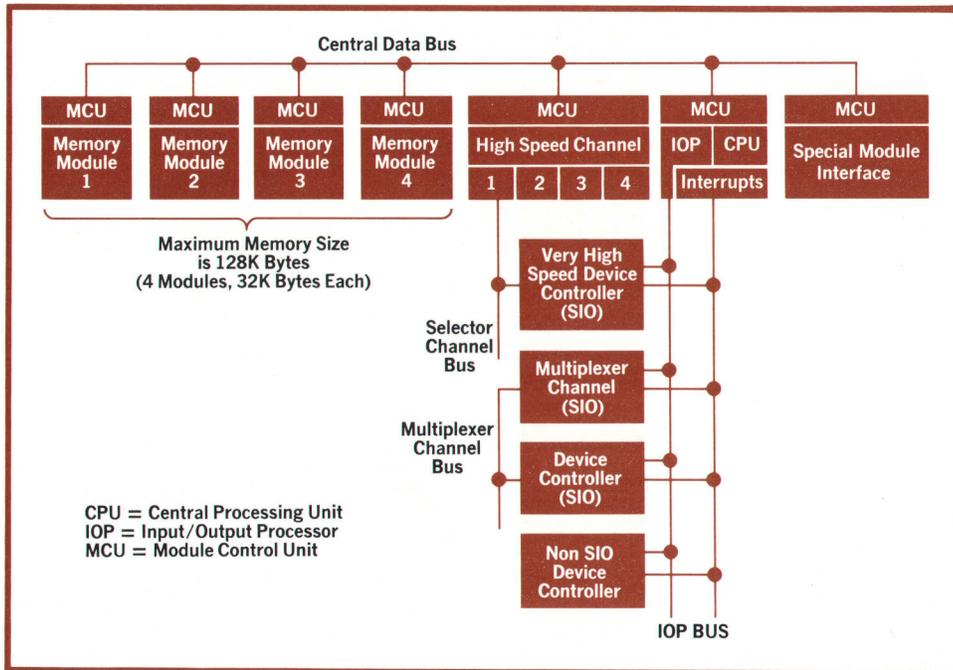
## **Bus Operation**

The central data bus is a high-speed synchronous bus that can service up to seven modules. The transfer cycle time of the central data bus is equivalent to the cycle time of the system master clock, 175 nanoseconds. During each transfer cycle sixteen bits of data plus parity and eight bits of source-destination addresses and operation code are transmitted from the source module to the destination module.

Control of the bus is distributed among the modules; there is no central control. The bus control and interface logic for a given module is in the module control unit (MCU) for that module.

Bus cycles are granted to a transmitting module when two conditions are met. First, the transmitting module must request a bus cycle from its MCU and the destination module must be willing to accept the message in the next cycle. The willingness of a module to accept a message is indicated by the logical state of its "Ready" line. There are seven "Ready" lines in the central data bus, one for each module.

The second condition that must be met before a module is granted a bus cycle is that there must not be any higher priority module seeking to obtain the next bus cycle. Module priority is a function of data transfer urgency. Memory modules have the highest priority, and the high-speed selector channel has a higher priority than the CPU or input/output processor (IOP). A module, when ready to transmit a message, blocks lower priority modules by lowering its "Enable" line. There are seven ded-



**Fig. 1.** Central data bus of HP 3000 serves up to seven independent modules. 16 bits of data plus parity and eight bits of address and operation code are transferred in 175 nanoseconds.

icated "Enable" lines, one for each module, in the central data bus. Each MCU checks the status of all higher priority modules prior to granting the next bus cycle to its host module.

With this bus-cycle allocation scheme, the "handshaking" mode of operation is not necessary, so data transfer speed is improved.

The central processing unit and the input/output processor share a module control unit. Thus the CPU and IOP share a single port on the central data bus. The IOP has a higher priority for bus access than the CPU, although both have independent access to the bus. The IOP provides the I/O devices with a direct path to memory through a buffered connection between the central data bus and the I/O bus.

### The Central Processor

Because it provides a great deal of instruction power very economically, the microprogrammed read-only memory (ROM) method of logic control was chosen for the HP 3000. The central processing unit, Fig. 2, has a general-purpose microprocessor structure with some special features to aid the stack architecture. The 170 individual instructions are implemented by sequences of microinstructions stored in the control ROM.

In the CPU are approximately 30 registers. Those of most interest to the user are the four top-of-stack data registers (A, B, C, D), three code-segment registers (PB, P, PL), a status register, an I/O mask register, an index register (X), and six stack pointer

registers (DL, DB, Q, SM, SR, Z). The DB register is the base of the stack, and the S register, defined as  $SM + SR$ , is the top of the stack. The area between Q and S is for local variables of the current procedure or routine. The top-of-stack registers are logical extensions of the stack area in core and their use greatly improves instruction execution time. The SR register tells how many of these registers are filled.

To improve the efficiency of handling data in the CPU, a two-stage "pipelined" data path structure is used. In the first stage, data is selected from the source registers and fed onto the two data buses (R and S) and into the bus storage registers shown in Fig. 2. These storage registers are the pipeline holding registers and serve as the data source for the second stage. In the second stage this data is processed through the arithmetic logic unit and a shift network, and the result is optionally tested and stored in selected destination registers. New data is entered into the stream on each clock pulse to keep the pipeline full and maximize throughput. The 175 ns clock time achieved with this structure is much lower than would have been possible if the whole source-to-destination processing were done in one clock period.

Communication paths from the CPU to outside modules include a path to memory through the MCU and central data bus, a path to device controllers through the I/O processor and I/O bus, and a path to the control panel through a special panel interface.

## CPU Operation

The CPU performs tasks by sequentially enabling the appropriate logic to pass data through the processing structure and to perform other non-data-path functions. For each sequential step a 32-bit ROM word, divided into seven coded control fields, enables the required functions. Each 32-bit ROM word constitutes a microinstruction. As shown in Fig. 3, the seven fields in each microinstruction are the R and S bus source register fields, the operation or function field, the shift field, the register store field, the test field, and a special field for executing non-data-related tasks.

Because each control field can, in general, select only one meaningful field option at a time, it was possible to encode them with little loss of capability. For a slight reduction in speed, field encoding, or "vertical microprogramming," offers considerable ROM cost savings over the one-bit-per-option method.

Branching capability is provided by redefining the R bus, shift, and special fields to be interpreted

as a branch address when a Jump or Jump Subroutine instruction occurs in the function field. Constants also are generated by redefining fields when a function field designator occurs.

## Programs and Microprograms

As the CPU executes a user program, it sequentially fetches software instructions from main memory. From the binary pattern of each instruction, a combination ROM lookup table and decoding logic generates a ROM address and stores it in a presettable indexing ROM address register. This register is used first to access and then to step through the sequence of microinstructions, or microprogram, that causes the software instruction to be executed. There is a microprogram in ROM for each of the 170 machine instructions.

The CPU executes a software program in the normal sequence of phases, that is, instruction fetch, data fetch, and execute. In the HP 3000 these phases are more accurately described as instruction pre-fetch, optional data address computation or hard-

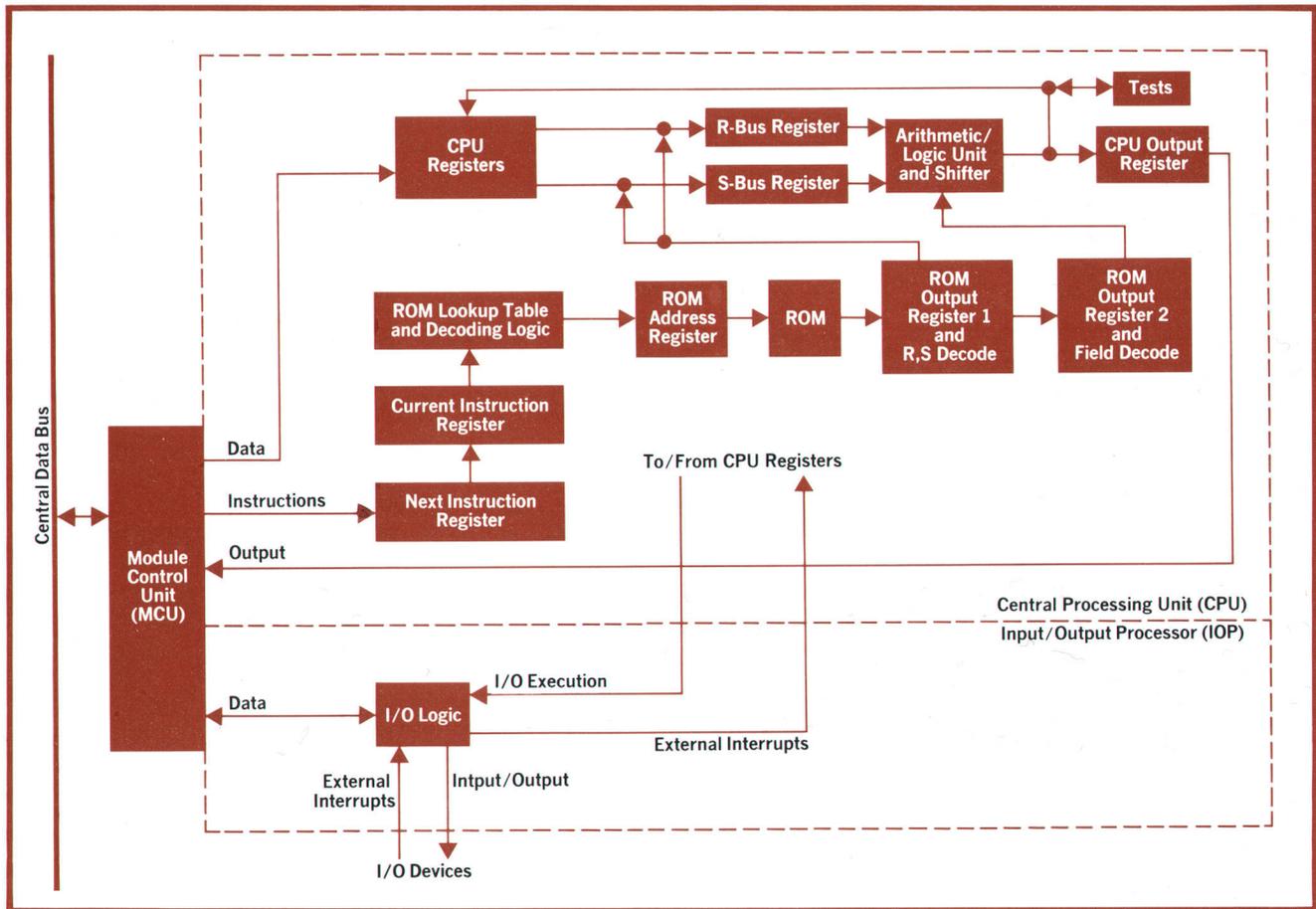


Fig. 2. Central processor has a general-purpose microprocessor structure with special features to aid stack operation.

ware stack register preadjust, and instruction execution. Instruction prefetch is an automatic hardware activity that gets the next instruction during the execution of the present instruction, thus avoiding the normal instruction fetch time. For memory reference instructions, hardware has been provided to compute the absolute memory address, that is, to add the displacement and index to the appropriate base register. A general bounds-testing routine in ROM then checks the computed address for validity before the individual instruction microprogram is used. Instructions that use only top-of-stack data normally (90% of the time) don't require a data fetch, but if necessary, these instructions are first routed through a microprogram that fills the appropriate number of hardware stack registers from the equivalent logical locations in core.

### **Interrupts**

As the execution of each instruction is completed a microprogram control signal is issued that starts the execution of the next instruction unless an interrupt is requesting service. If an interrupt has occurred, a force to an interrupt microprogram takes place. This causes the status of the present user program to be stored on the stack. Then if the interrupt is not directly user related, the microprogram transfers the status to a system interrupt stack and calls the first instruction of the software program serving that interrupt. After the interrupt has been serviced control is returned to the MPE operating system.

### **TOS Hardware**

To achieve faster execution of instructions that reference the top elements of the stack, special hardware has been provided. Up to four of the top elements of the stack can be kept in four top-of-stack hardware registers, and manipulation of these registers by the microcode has been made as easy as possible. The TOS hardware includes the four registers and renaming logic that allows each of the four registers to assume any of the four positions relative to the top of the stack. Thus, the stack can be logically shifted up or down by simply renaming the registers, without moving the contents of one register to another. The number of stack elements that currently reside in the TOS hardware registers is kept in the TOS register pointer, SR.

### **Memory**

Memory modules on the HP 3000 are designed to be self-contained asynchronous units of up to 64K bytes each. The maximum memory limit is 128K bytes in up to four modules. The modules interface with the system through an MCU port on the cen-

tral data bus. Only data transmissions to the system have to be synchronized with the system clock; all other memory timing and control is contained within each module. Since no fixed response time is required, faster memories can be interfaced as they become available.

Memory commands include read, write, and a special multiprocessor semaphore function: read and write all 1's within one memory cycle.

The present memory is a 960-nanosecond three-wire 3D magnetic core memory using the same core stack and phased X-Y drive current arrangement as is used in the HP 2100A Computer.<sup>1</sup> A basic module consists of one timing, control, and MCU interface card, one X-Y switch and inhibit-current load card and one to four 8K word stack cards. Because the sense amplifiers, X-Y drivers and inhibit drivers all are on the stack card, memory expansion only requires the addition of one stack card for each additional 16K bytes.

### **Input/Output Processor**

The functions of the I/O processor have been distributed between a kernel processor attached to the CPU and one or more multiplexer channels on the I/O bus. The kernel processor controls the I/O bus, which is the data path from external devices to memory and the communication path between external devices and the CPU. The multiplexer channel does the bookkeeping for block transfers of data to and from memory for up to 16 devices. When needed, additional multiplexer channels may be added to the system.

Input/output operations in the HP 3000 are divided into three categories: direct I/O, programmed I/O and interrupt processing. Programmed I/O operations have priority on the I/O bus over other types.

Direct I/O operations take place as a result of the execution of an I/O instruction by the CPU. These operations either exchange a word of information between the top-of-stack register (TOS) in the CPU and the I/O device controller, or cause a control function to take place in the I/O system. During the execution of I/O instructions the CPU microprocessor performs the basic control functions such as assembling the I/O command, checking the status of the I/O device controller, and exchanging a word of information between the TOS register and the I/O device via the I/O bus.

Programmed I/O operations are aimed at transferring blocks of data between I/O devices and the memory. This type of operation begins for an I/O device when the CPU issues an SIO instruction for that device. The device controller in conjunction



## SPECIFICATIONS HP 3000 Computer System

### DESCRIPTION

Multiprogrammed general-purpose computer system implemented with complementary hardware and software providing for concurrent real-time, batch, and time sharing processing.

### CENTRAL PROCESSOR

#### ARCHITECTURE

Hardware-implemented stack  
Separation of code and data  
Nonmodifiable, reentrant code  
Variable-length code segmentation  
Virtual memory  
Dynamic relocatability of programs

#### IMPLEMENTATION

Microprogrammed CPU  
175 nanosecond microinstruction time  
Memory protect, parity checking, power-fail/auto restart  
Protection between users  
Central data bus  
Concurrent I/O and CPU operations

#### INSTRUCTIONS

170 instructions  
16 bits per word  
16-bit and 32-bit integer; 32-bit floating point hardware arithmetic  
Triple-word shifts to aid 48-bit floating point software

#### MEMORY

Technology independent, speed independent  
Up to four modules  
Interleaving  
Addressable to 65K words (131,072 bytes)  
17 bits includes parity bit

#### I/O AND PERIPHERALS

##### GENERAL

Privileged control of I/O

#### Concurrent I/O operations

Three ways to implement I/O  
Direct memory access by all channels  
Device-independent I/O program execution  
Up to 253 devices

#### I/O SYSTEM

Multiplexer channel  
Selector channel  
Direct I/O

#### INTERRUPT SYSTEM

Up to 253 external interrupts  
Independent masking and priority structures  
Microprogrammed environment switching  
Common stack for interrupt processing  
17 internal interrupts plus 7 traps

#### PERIPHERALS

Mass Storage:  
Fixed Head Disc: 1, 2 or 4 megabyte, 496 kHz byte transfer rate  
Removable Media: 47 megabytes, 320 kHz byte transfer rate  
4.9 megabytes, 245 kHz byte transfer rate  
Magnetic Tape: 7 Track—45 ips, 200, 556 or 800 bpi  
9 Track—45 ips, 800 or 1600 bpi  
Card Readers: 600 or 1200 cards per minute  
Card Punches: 35 or 250 cards per minute  
Line Printer: 200 or 600 lines per minute, 64 or 96 character sets; 132 columns  
Paper Tape Reader: 500 cps  
Paper Tape Punch: 75 cps  
Consoles: CRT or ASR-33

#### SOFTWARE

##### MULTIPROGRAMMING EXECUTIVE

Batch processing  
On-line terminal processing  
Real time processing  
Production control  
Automatic testing

#### Process Control

Information retrieval  
Data acquisition

#### SYSTEMS PROGRAMMING LANGUAGE (SPL)

High-level syntactic structure  
Ability to address hardware registers explicitly  
Bit manipulation  
Branches based explicitly on hardware status  
Use of all hardware data types and operators

#### PROGRAMMING LANGUAGES

FORTRAN: extended version of ANSI Standard FORTRAN (x3.9-1966)

BASIC: most powerful currently available  
COBOL: highest level of Federal Government Standard  
Compiler Library: provides common compiler functions

#### DATA MANAGEMENT

IMAGE: comprehensive information management system  
QUERY: interactive language interface to data base via IMAGE

#### SUPPORT SOFTWARE

EDIT: editing of source or text files  
SORT: sort and/or merge of multiple files  
TRACE: debugging tool for FORTRAN and Systems Programming Language

#### SCIENTIFIC SOFTWARE

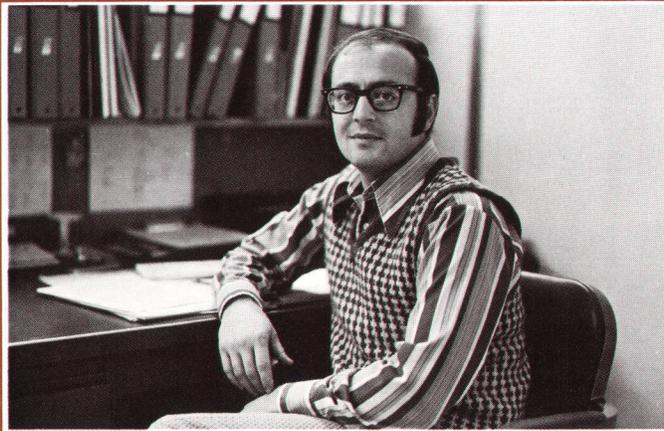
Scientific Library  
STAR—Statistical Analysis Routines

#### DIAGNOSIS OF HARDWARE

System Diagnostic Monitor—runs on-line diagnostics  
Stand-alone diagnostics  
Microdiagnostics

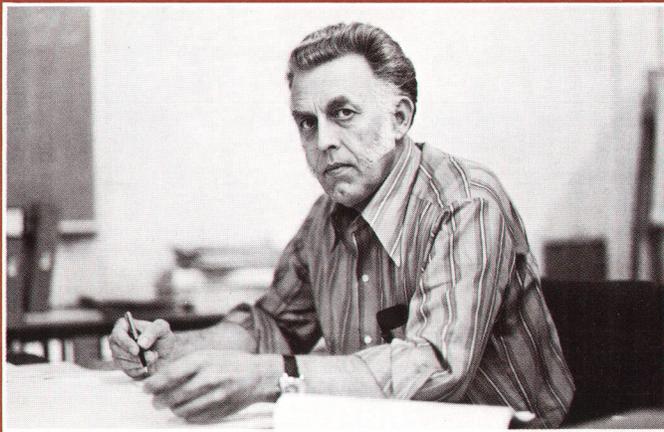
**PRICE IN USA:** \$125,000 for small system to over \$500,000 for large system.

**MANUFACTURING DIVISION:** HP Data Systems  
11000 Wolfe Road  
Cupertino, California 95014



### Jamshid Basiji

Since coming to HP in 1969, Jim Basiji has worked on high-speed I/O processing techniques, developed the architecture of the I/O system and central data bus for the HP 3000, and helped design the I/O processor and central processor modules for the HP 3000. A graduate of the University of California at Berkeley, Jim received his B.S.E.E. degree in 1965 and his M.S.E.E. degree in 1966. Before joining HP, he worked on computer development and advanced computing techniques for IBM. His idea of a fascinating way to spend his free moments is with a good book.



### Arndt B. Bergh

With HP since 1956, Arne Bergh has had a variety of research and development responsibilities as a member of HP Laboratories and various operating Divisions. His projects have included instruments, magnetic devices, memories, and computers, the latest being the hardware design of the HP 3000. He holds four patents and has others pending on the HP 3000 and on a 1024-bit bipolar ROM. Arne received the A.B. degree in chemistry from St. Olaf College in 1947 and the M.S. degree in physics from the University of Minnesota in 1950. He's a member of ACM and IEEE. He has a private pilot's license, but his real passion is flying over the water, racing his Daysailer sailboat in local, regional and national competition.

# Software for a Multilingual Computer

*SPL is a high-level language that produces code that's as efficient as other systems' assembly-language code. Other 3000 languages are FORTRAN, BASIC and COBOL.*

by William E. Foster

**P**ROGRAMMING LANGUAGES NOW AVAILABLE FOR THE HP 3000 USER are FORTRAN, BASIC, and SPL (Systems Programming Language). COBOL will be available in summer 1973. The system will support all these languages simultaneously.

## Systems Programming Language

SPL is an ALGOL-like language. Its objective is to provide systems programming capability from a high-level language rather than the traditional assembly language. The benefits are faster coding and easier debugging. Virtually all the HP 3000 software is written in SPL.

It's imperative, of course, that a systems programming language produce efficient object code, and this was another major objective of SPL. Code optimization has been achieved through the logic of the compiler and through close correlation between the SPL syntax and the 3000 instruction set.

A significant aspect of SPL is that it may be used as either a machine independent or a machine dependent programming language. At the machine independent level, the syntax of SPL closely resembles that of ALGOL. It isn't necessary for the programmer to understand the architecture of the 3000 to program at this level.

The machine dependent programmer is one who has some knowledge of the 3000 architecture (instruction set, stack, status register, etc.); the greater his knowledge, the more he is able to make use of the machine dependent features of SPL. The effect of using these features of SPL is improved object code.

Fig. 1 illustrates the two levels of SPL applied to the same programming problem. Fig. 2 is an example of a more typical SPL program.

## FORTRAN/3000

FORTRAN is one of the most widely used and oldest programming languages. Initial specifications for the language date back to 1954. FORTRAN/3000 is an ANSI-standard compiler with extensions that enhance the capability of the language and use the features of the HP 3000. Among these features are CHARACTER variables, which were added to the language to provide the capability of string manipulation. Additionally, a great deal of power is provided in the area of input/output operations.

- Free-field I/O. Variables may be input and output in a free-field manner, without the specification of a FORMAT statement.
- Output expressions. Expressions may be included in the output list (Fig. 3). For example,  

```
WRITE (3,10) I*S,A+B
```

is a legal FORTRAN/3000 statement.
- Logical unit table. A global table is created by the compiler and built by the loader that is used to associate FORTRAN logical unit (storage device) numbers with internal file numbers. The FORTRAN programmer has the capability, with the use of library routines, to tailor this table to his own needs. For instance, he may explicitly open a file through a call to the file system intrinsic FOPEN, then set the returned file number to correspond to a particular FORTRAN unit number (say unit #7). Subsequent READ or WRITE statements using unit #7 would, in fact, be referencing this file.
- FORMAT specification. Two important specifications have been added to the FORMAT statement: the T-specification, which positions the format scanner to specific locations in the record, and the S-specification, which outputs character data with a field width that corresponds to the

### Machine independent method

The conventional approach, used in most programming languages, would be to use a temporary variable in making the exchange:

| SPL statement | Purpose                             | Generated Code      |
|---------------|-------------------------------------|---------------------|
| TEMP = A;     | Store the value of A in TEMP.       | LOAD A<br>STOR TEMP |
| A = B;        | Store the value of B in A.          | LOAD B<br>STOR A    |
| B = TEMP;     | Store the original value of A in B. | LOAD TEMP<br>STOR B |

### Machine dependent method

A more efficient approach would be to use the special SPL symbol TOS. When used in place of an identifier, this symbol denotes the current top of stack.

| SPL statement | Purpose   | Generated Code   |
|---------------|---|------------------|
| TOS = A;      | Push the value of A onto the stack.   | LOAD A           |
| A = B;        | Store the value of B in A.  | LOAD B<br>STOR A |
| B = TOS;      | Store the current value that is on the top of the stack into B, then pop the stack. | STOR B           |

Fig. 1. An SPL program to swap the values of two integer variables, A and B, illustrating the machine dependent and machine independent levels of the language.

length of the associated list element (Fig. 3).

- Direct-access I/O. Disc files may be referenced as direct access devices. For example, the statement

```
READ (3@RECNUM) A,B
```

reads from logical unit #3 the record specified by RECNUM, and transmits the data to the list elements A and B.

Other extensions of standard FORTRAN are mixed-mode arithmetic, free format program entry for more convenient usage from terminals, removal of restrictions on indexing and DO-loops, and an interactive debugging facility.

Machine dependent characteristics of FORTRAN/3000 are that programs are recursive and reentrant. In the HP 3000, code and data are stored separately, and code is never altered. This means that programs can be shared by several jobs. If one job is using a program and is interrupted by another job that uses the same program, the first job can later reenter the program and continue from the point of interruption. Thus programs are reentrant. Recursive means that programs can call themselves. Another machine dependent feature is that storage for local variables is allocated on the stack dynamically when functions or subroutines are entered, and deallocated upon exit.

### SCIENTIFIC LIB SYMMETRIC MATRIX INVERSION ROUTINE

```

01777000 0000 1  SCONTROL SEGMENT = S'LIB'11
01778000 0000 1  PROCEDURE SMETINVA(N,N,F) <<INVERT SYMMETRIC MATRIX A OF ORDER N>>
PL=27
01779000 0000 1  VALUE N1
01780000 0000 1  REAL ARRAY A1
01781000 0000 1  INTEGER N1
01782000 0000 1  LOGICAL F1
01783000 0000 1  BEGIN
01784000 0000 2  REAL BIGDIAG*ATOS*5=11
0 + #01
0 + #01
01785000 0000 2  INTEGER I+J*K*L*M*XXKI
0 + #03
0 + #04
0 + #05
0 + #06
0 + #07
0 + #08
01786000 0000 2  REAL ARRAY P(BIN)+Q(BIN)1
0 + #10
0 + #11
01787000 0000 2  LOGICAL ARRAY R(BIN)1
0 + #12
01788000 0000 2  LOGICAL POINTER RP(N)1
0 + #12
01789000 0000 2  DEFINE ZERO = 0. #1
01790000 0000 2  DEFINE ONE = 1. #1
01791000 0000 2  COMMENT START MAIN PROCEDURE *****
01792000 0000 2  F = FALSE
01793000 0000 2  N = N-1
01794000 0025 2  RP=TRUE1
01795000 0027 2  TOS = RP+11
01796000 0030 2  MOVE *RP+(N)1 <<INITIALIZE R TO TRUE>>
01797000 0034 2  FOR I = 0 UNTIL N DO <<GRAND LOOP>>
01798000 0041 2  BEGIN
01799000 0043 3  BIGDIAG = ZERO1
01800000 0045 3  L = -11
01801000 0047 3  FOR J = 0 UNTIL N DO <<SEARCH FOR PIVOT>>
01802000 0054 3  BEGIN
01803000 0060 4  L = L + J+11
01804000 0064 4  IF R(J) THEN
01805000 0067 4  BEGIN
01806000 0067 5  TOS = A(L)1
01807000 0067 5  IF < THEN ASSEMBLE(FNEG)1
01808000 0073 5  IF ATOS > BIGDIAG THEN
01809000 0077 5  BEGIN
01810000 0080 6  BIGDIAG = TOS 1
01811000 0086 6  K = J1 + L1
01812000 0094 6  END ELSE ASSEMBLE(ODEL)1
01813000 0096 6  END1
01814000 0106 4  END1 <<SEARCH FOR PIVOT>>
01815000 0107 3  IF BIGDIAG = ZERO THEN
01816000 0111 3  BEGIN
01817000 0113 4  F = TRUE1
01818000 0115 4  RETURN1
01819000 0126 4  END1
01820000 0128 3  COMMENT PREPARATION OF ELIMINATION STEP II
01821000 0128 3  TOS = ONE/A(N)1
01822000 0128 3  A(K) = ZERO1
01823000 0128 3  Q(K) = TOS1
01824000 0130 3  P(K) = ONE1
01825000 0132 3  R(K) = FALSE1
01826000 0134 3  L = M - K1
01827000 0137 3  FOR J = 0 UNTIL K-1 DO
01828000 0145 3  BEGIN
01829000 0152 4  TOS = Q(K)1
01830000 0157 4  TOS = A(L)1 <<A(J,K)>>
01831000 0161 4  A(K) = ZERO1
01832000 0163 4  P(J) ATOS1
01833000 0166 4  IF R(X) THEN ASSEMBLE(FNEG)1
01834000 0171 4  ELSE ASSEMBLE(FMPY)1
01835000 0175 4  Q(X) = TOS1
01836000 0176 4  L = L + 11
01837000 0177 4  END1
01838000 0180 3  FOR J = K+1 UNTIL N DO
01839000 0205 3  BEGIN
01840000 0218 4  L = L + J1
01841000 0219 4  TOS = A(K)1
01842000 0219 4  TOS = A(L)1 <<A(K,J)>>
01843000 0221 4  A(K) = ZERO1
01844000 0221 4  IF R(J) THEN TOS = ATOS
01845000 0224 4  ELSE TOS = -ATOS1
01846000 0231 4  P(K) = TOS1
01847000 0231 4  ASSEMBLE(FNEG)1 <<A(K,J)*Q(K)>>
01848000 0234 4  Q(X) = TOS1
01849000 0234 4  END1
01850000 0235 3  COMMENT ELIMINATION PROPER1
01851000 0235 3  L = 01
01852000 0237 3  FOR K = 0 UNTIL N DO
01853000 0244 3  FOR J = 0 UNTIL K DO
01854000 0253 3  BEGIN
01855000 0255 4  A(X) = P(J)*Q(K) + A(L)1 <<A(L)*A(J,K)>>
01856000 0266 4  L = L + 11
01857000 0267 4  END1
01858000 0271 3  END1 <<GRAND LOOP>>
01859000 0272 2  END1 <<SMETINV>>
IDENTIFIER CLASS TYPE ADDRESS
A ARRAY REAL 0 -005
ATOS SIMP. VAR. REAL S -001
BIGDIAG SIMP. VAR. REAL 0 -001
F LOGICAL 0 -004
I SIMP. VAR. INTEGER 0 -003
J SIMP. VAR. INTEGER 0 -004
K SIMP. VAR. INTEGER 0 -005
L SIMP. VAR. INTEGER 0 -006
M SIMP. VAR. INTEGER 0 -007
N SIMP. VAR. INTEGER 0 -005
ONE DEFINE 0 -005
P ARRAY REAL 0 -010
Q ARRAY REAL 0 -011
R ARRAY LOGICAL 0 -012
RP POINTER LOGICAL 0 -012
X SIMP. VAR. INTEGER XRG
ZERO DEFINE 0.
00000 035013 171700 051410 041605 003300 010001 035000 171700 00010 051411 041605 003
00000 003300 035000 000000 053004 121605 025001 053412 041412 00030 003300 041412 041
00000 041605 050002 140230 034013 161401 025001 051406 000600 00050 051404 171404 021
00000 041406 071406 003300 051406 131404 047412 013720 131406 00070 157606 141602 005
00000 041406 051405 041406 051407 140002 000200 052426 151401 00110 034006 005000 141
00120 034027 131407 157606 005400 034025 167606 131405 167411 00130 034022 167410 000
00100 051404 171404 021001 041405 003400 050010 140032 040000 00150 000000 000000 000
00260 157606 034011 157606 004000 131404 167410 047412 013705 00170 005553 140004 000
00200 041405 003300 051404 171404 021001 041605 050002 140020 00210 041406 071404 051
00220 167606 131406 047412 013705 004600 140004 000000 000000 00230 004655 167410 005
00240 051405 171405 021001 041605 050002 140004 000000 051404 00250 171404 021001 041
00260 157411 005300 131406 157606 005100 167606 120406 052412 00270 052422 052626 031

```

Fig. 2. A matrix inversion routine from the HP 3000 Scientific Library, written in SPL. The compiler output shown here includes much optional information (shown in color), such as sequence numbers, PB-relative address of source statements, a BEGIN/END code dump, a symbol table dump, and a machine code dump.

## BASIC/3000

The HP 3000 BASIC subsystem runs as an interpreter rather than a compiler, which means that programs are not translated into machine code that is directly executable, but into an intermediate language that is executed by control routines.

The primary reasons for having an interpreter instead of a compiler are faster development and greater debugging facilities. The interactive debugging mode in BASIC provides the following capabilities:

- Tracing of the path of execution through a program and changes in the values of variables
- Interactively displaying the dynamic nesting structure of a program, that is, the order in which programs and functions are called
- Displaying and modifying the values of variables
- Altering the execution sequence of a program.

One aspect of an interpreter is that programs are really data to the interpreter. Therefore, BASIC programs do not execute as code segments and so are not sharable. For this reason, HP is currently developing a BASIC compiler that accepts the internal file generated by the interpreter and generates executable code. In this way, BASIC programs will not only run as sharable code segments, but will also execute faster.

The BASIC/3000 language is a superset of HP 2000 BASIC, incorporating many extensions:

### 2000

26 numeric arrays  
26 string variables  
one data type (32-bit real)

### 3000

286 numeric arrays  
286 strings or string arrays  
four data types (16-bit integer, 32-bit real, 48-bit real, 64-bit complex)

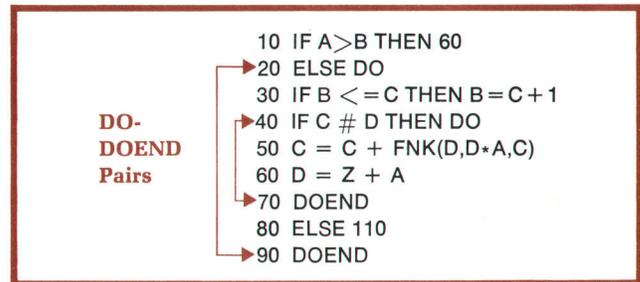
Other extensions include compound statements (Fig. 4), mixed-mode arithmetic, multiple-line functions, string-valued functions, access to all MPE

```
WRITE(6,10) "PRESSURE", P
WRITE(6,10) "TEMPERATURE", 2*T
10 FORMAT (" THE VALUE FOR ", S, " IS", F7.3)
```

Result: (assume P=1.0339 and T=55.87)

```
THE VALUE FOR PRESSURE IS 1.034
THE VALUE FOR TEMPERATURE IS 111.740
```

**Fig. 3.** FORTRAN/3000 program illustrating the use of an expression in an output list, and the "S" specification in the FORMAT statement.



**Fig. 4.** An example of a BASIC/3000 compound statement.

files and peripheral devices, capability of calling SPL procedures, many additional predefined string and numeric functions, string arrays, program overlays, picture I/O formatting, statement execution frequency reporting, dynamic array redimensioning, handling of non-BASIC files, and additional file commands.

SPL, BASIC, and FORTRAN are all recursive, that is, programs, procedures, and subroutines can call themselves. Fig. 5 illustrates this property.

## COBOL/3000

COBOL (COmmon Business Oriented Language) is the result of an effort to establish a standard programming language for business processing. The original specifications were drawn up in 1959 by CODASYL (the COmference on DATA SYstems Languages). COBOL/3000 conforms to the highest level of Federal Government Standard COBOL and has the added capability of interprogram communications.

COBOL is a structured language that consists of Identification, Environment, Data, and Procedure divisions. A feature of COBOL that makes it attractive in commercial applications is that it provides fixed-point arithmetic up to 18 digits; this eliminates the problem of round-off error which exists in "floating-point" formats.

## Switching Languages Made Easy

HP 3000 languages share many common attributes that aid the user in switching from one language to another. Among the areas of compatibility are:

- Program-to-program communication. SPL, FORTRAN, and COBOL programs can all call programs written in either SPL, FORTRAN, or COBOL. BASIC programs can call SPL, FORTRAN, or COBOL programs as well as other BASIC programs. Files written in one language are accessible by other languages.
- Compiler construction. The command languages for all of the compilers are consistent. For ex-

**SPL**

```

INTEGER PROCEDURE FAC (N); VALUE N; INTEGER N;
FAC := IF N <= 1 THEN 1 ELSE N * FAC (N - 1);

```

**FORTTRAN**

```

INTEGER FUNCTION FAC (N)
IF (N. GT. 1) GO TO 10
FAC = 1
RETURN
10 FAC = N * FAC (N - 1)
RETURN
END

```

**BASIC**

```

100 DEF INTEGER FNF (INTEGER N)
110 IF N <= 1 THEN RETURN 1
120 ELSE RETURN N * FNF (N - 1)
130 FNEND

```

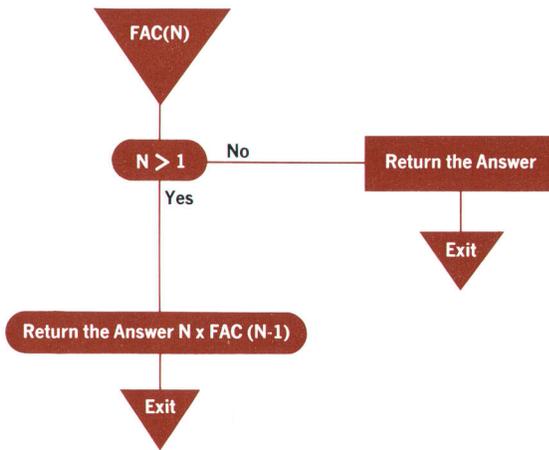


Fig. 5. SPL, BASIC/3000, and FORTRAN/3000 programs to calculate integer factorials. All three languages have recursive capabilities.

ample, the commands that tell the compiler to merge a source file with an update file are identical for each compiler. Also, the language translators share the same system library routines. These library routines are used both during compilation and as run-time routines to implement the language features. For example, the program that converts a character string into an internal binary number is used both by SPL at compile time and by the FORTRAN formatter at execution time. This modularity not only simplifies the task of making changes to common programs, but also reduces the development cost by eliminating duplication of effort. The steps in compiling and executing programs are as follows (Fig. 6):

- 1) The source program (main program plus sub-

outines) is compiled into relocatable modules that are stored in the user's subprogram file (USL). If the programmer decides to change any part of his program, he can recompile any subroutine, or the main program, into the USL file and the old copy of that subroutine will be deactivated. (It will still exist in the file, and could later be reactivated.) The relocatable modules can be added, deleted, activated, or deactivated from the USL. Also, these modules can be copied from one USL to another.

- 2) Next, the USL file is prepared into a Program File. Preparation consists of segmenting the code and defining the initial stack size.
- 3) Now, the Program File can be allocated/executed. The segments are allocated into virtual memory, external references are satisfied from the libraries, and the program is scheduled for execution according to its priority.

**General-Purpose Applications Software**

Several general-purpose software packages are now available for the HP 3000. There is a scientific library, an interactive statistical package, a text editor, and a text formatter. Other packages will be available in the future.

**Scientific Library.** The scientific library consists of a collection of SPL procedures that reside in the system library. The initial capabilities include: error function/complimentary error function, gamma and log<sub>e</sub> gamma functions, exponential, sine-cosine, Fresnel integrals, elliptic integrals and elliptic functions, Bessel functions, and statistical procedures including elementary statistics (kurtosis, means, etc.), one-way frequency distribution, correlation, and multiple linear regression. This library

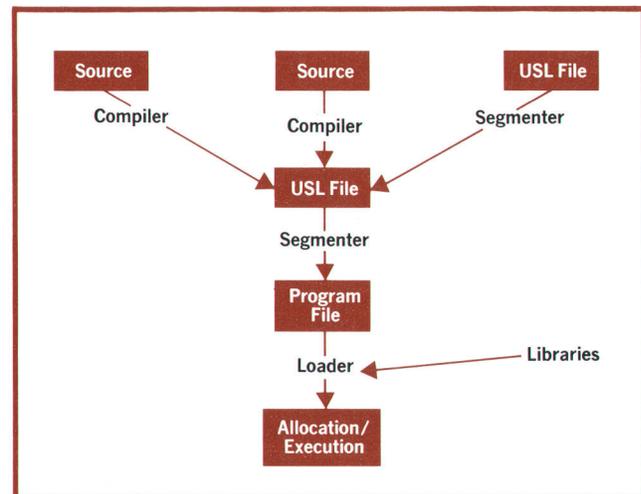


Fig. 6. HP 3000 compilation/execution process.

will be kept open for future enhancement.

### **Interactive Statistical Analysis Package (STAR).**

This subsystem provides the user with the capability of performing various kinds of statistical analysis in an interactive (question-answer) mode. This package may also be used in a batch mode. All of the statistical capabilities that exist in the scientific library are available to the STAR user, along with the following additions: data file manipulation (creation, editing, etc.), scatter diagrams, histograms, and variable transformation.

The output from STAR may be to the user's terminal, or to a line printer. All results are displayed in an easily readable, tabular form. The data may be input directly from the terminal, or from the batch input device, or from a file created by a FORTRAN, SPL, or BASIC program.

In keeping with the modular structure of the HP 3000 system, STAR makes use of the scientific and compiler libraries in performing its functions. As new capabilities are added to the scientific library, these capabilities will be easily extendable to STAR merely by adding the necessary input/output routines and calling on the scientific library to perform the calculations.

**Text Editor.** EDIT/3000 is a general-purpose utility that provides the user with the capability of easily creating and manipulating files of upper and lower case ASCII characters. Lines and characters can be inserted, deleted, replaced, searched for, and so on. The files to be edited can be FORTRAN, SPL, BASIC, or COBOL source files, or textual material such as reports.

One feature of this program not usually found in text editors is its ability to selectively modify text depending on conditions found within the text itself. When this is done, the "edit language" has an ALGOL-like structure with the metacommands WHILE, NOT, and OR acting upon statements that can be compound statements (groups of statements enclosed by a BEGIN-END pair). These commands and statements can be nested indefinitely. Interactive users can write an edit program to send messages to the terminal and place input from the user in appropriate places within the text file. Together, these features make the editor a powerful tool for many applications other than simple program editing.

**Text Formatter.** This program lists ASCII files under the control of format records imbedded in the text file. FORMAT/3000 may also be used with the text editor. The formatter provides the capability of preparing simple documents to be listed on line

printers or other ASCII devices.

### **Acknowledgments**

The following people were directly involved in the implementation of the languages and general-purpose products:

SPL: Doug Jeung, Gerry Bausek, Tom Blease.

FORTRAN: Jerry Smith, Terry Hamm, Jim Hewlett, John Couch

BASIC: Mike Green, Terry Opdendyk, John Shipman

COBOL: Steve Ng, Waldy Haccou, John Welsch, John Yu, Paul Rosenfeld, Gerry Bausek

STAR, Scientific Libraries: Paul Rosenfeld, Dave Johnson

Editor/Formatter: Fred Athearn.

Credit is also due the many people in software QA and publications who have done such a great job. 



### **William E. Foster**

As section manager for systems software, Bill Foster is responsible for programming languages and operating systems for 2100, 2000, and 3000 Computer Systems. Bill received his B.A. degree in mathematics from California State University at San Jose in 1966, then spent the next three years developing satellite orbit prediction, tracking, and reentry software. In 1969 he got his M.S. degree in applied mathematics from the University of Santa Clara and joined HP as a software project manager. He became a section manager in 1971 and assumed his present job in 1972. A member of ACM and the American Management Association, Bill is now a candidate for the M.B.A. degree at Santa Clara. He enjoys golf, tennis, bicycling, basketball, hydroplaning (he built his own boat), and exploring the Bay Area by motorcycle, and he's now taking flying lessons.

# Single Operating System Serves All HP 3000 Users

*The Multiprogramming Executive operating system takes care of command interpretation, file management, memory management, scheduling and dispatching, and input/output management for time-sharing, batch, and real-time users.*

by Thomas A. Blease and Alan Hewer

**M**ULTIPROGRAMMING EXECUTIVE (MPE/3000) is a general-purpose disc-based operating system that supervises the operation of the HP 3000 Computer System and its variety of users.

MPE/3000 allows users to access the system concurrently in three distinct but compatible modes: batch processing, time sharing, and real-time processing. MPE is designed to take maximum advantage of system resources, to make the system easy to use, and to relieve the user of the need for detailed knowledge of the internal hardware or direct interaction with it. Each user's environment is protected; program protection is provided by hardware and data protection by any of several software facilities depending on the degree of security desired.

MPE/3000 has a modular organization that makes it more convenient to check out and maintain, and provides a flexible base on which additional capabilities may later be developed. Users interact with the 3000 System through the command interpreter, one of the functional units of MPE. Programming access to the hardware is provided by system routines called MPE intrinsics. Uniform access to disc files and input/output devices is provided by the MPE file system. MPE also has memory management, an input/output system, and scheduling for dynamic allocation of resources.

## Process Structure

Underlying the modularity of MPE/3000 and its ability to support three kinds of users concurrently is its process structure. Except for a few specialized system controls such as the dispatcher and interrupt structure, all operating-system and user functions are performed as a series of processes.

A process is the basic entity that can be executed by the central processor. While a program identi-

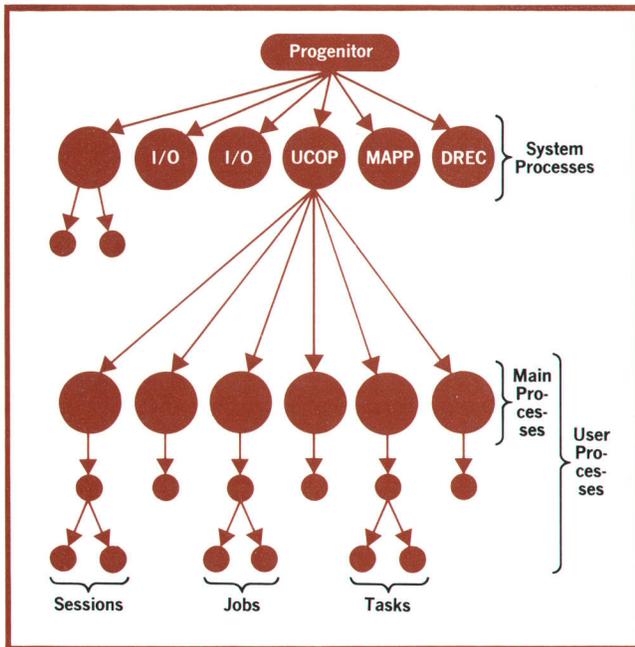
fies a static sequence of instructions and data, a process denotes the dynamically changing sequence of states of an executing program. Under MPE/3000, a process consists of:

- A unique process control block which describes and controls the process,
- A private (stack) data segment, accessible only by the process, for data operation and storage, and
- An instruction in a code segment which may be private to the process or may be shared with other processes.

Processes are organized hierarchically in a tree structure as shown in Fig. 1. Each process has only one immediate ancestor, but may have several immediate descendants. Control and information flows are restricted to proceed only along branches of this logical tree structure. The primary interactions which are provided for are creation, deletion, control, and intercommunication.

The root process is the *progenitor*. All immediate descendants of the progenitor are *system processes*. They include:

- I/O system controller processes, which queue, initiate, and complete all input/output requests for all devices configured under the operating system.
- The make-a-process-present (MAPP) process, which schedules the allocation of memory resources to data segments belonging to active processes.
- The device recognition (DREC) process, which performs the administrative tasks of allocating input/output devices and also verifying and initiating new users under the operating system.
- The user controller (UCOP) process, which is



**Fig. 1** Multiprogramming Executive (MPE) operating system for HP 3000 has a process structure. All functions are performed as a series of processes.

defined as the ancestor of all user processes currently running under MPE/3000. The primary responsibility of UCOP is to create, supervise, and delete user process tree structures.

Of these system processes, the most important is UCOP, the root process of the user structure. An immediate descendant, created by UCOP, is called a *main process*, and the code executing under it is normally the command interpreter. The process tree structure originating at a main process defines a *job* (job/session/task). A basic feature of a job is its complete independence from all other jobs currently existing.

Apart from the progenitor and several specific system processes which together constitute the operating system and which must exist, the process tree structure is completely dynamic, expanding and contracting as operating system and user requirements change.

### Memory Management

The primary function of MPE/3000 memory management is the allocation of main memory to meet the demands of users. "Main memory" is core memory as opposed to disc memory. The memory management module is also responsible for code segment table entries, data segment table entries, and overlay disc storage for data segments.

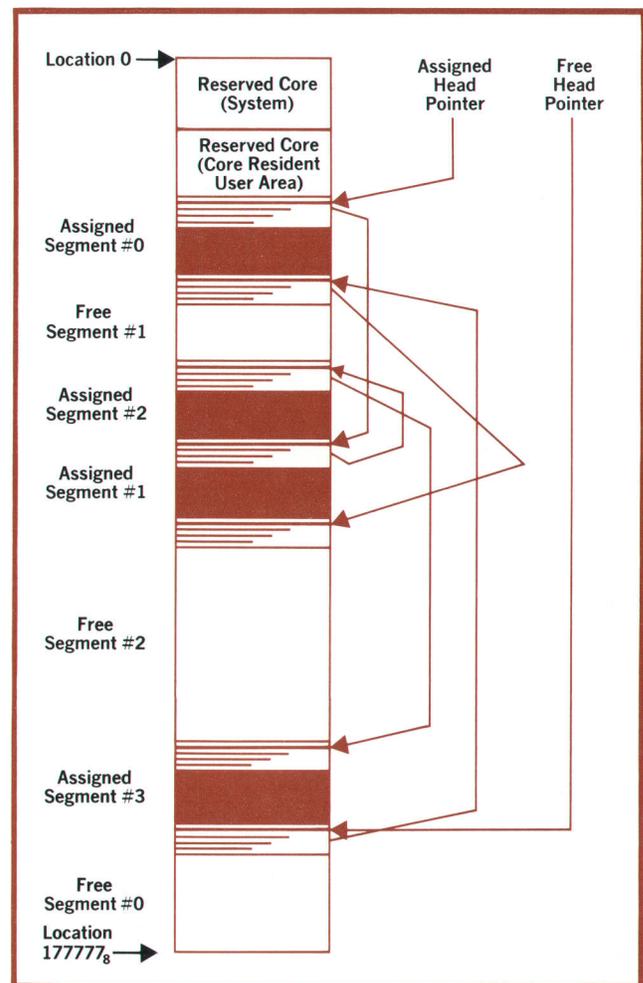
### Main Memory Organization

Main memory is organized into three contiguous

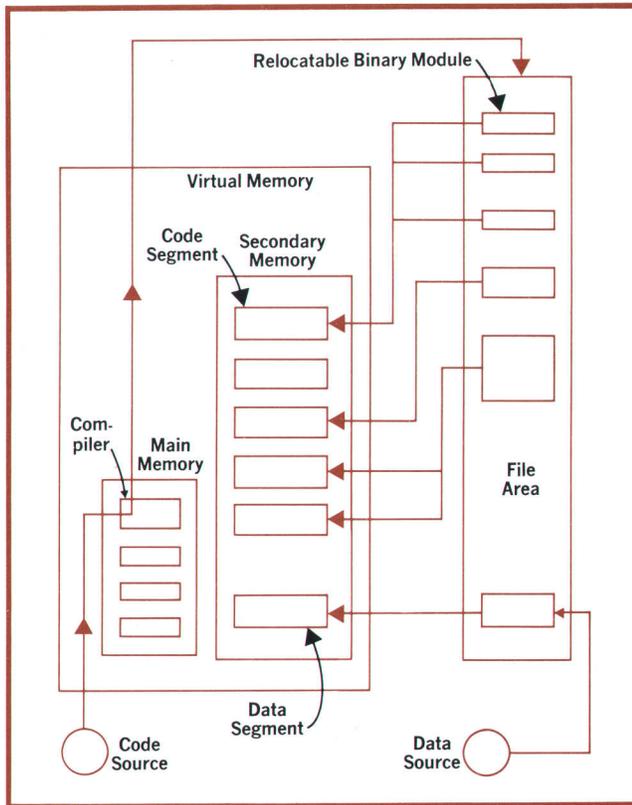
areas (Fig. 2). The first area contains system tables, interrupt procedures, and MPE intrinsics which must be core resident, that is, always present and accessible in main memory.

The second area is of variable length and is used to satisfy requests from users for core resident storage. This area is dynamically expanded and contracted and can be of zero length.

The remaining main memory is referred to as *linked memory*. Linked memory is composed of free (not currently being used) and assigned (allocated for a code or data segment) areas of varying sizes. Areas not currently in use are linked together and form the free space list. Similarly, the assigned areas are linked together and form the assigned space list. Each area contains an information header defining its size. If the area is assigned, the header also contains information about disposition (I/O pending, etc.), segment type (code or data with in-



**Fig. 2** Main memory is organized into reserved and linked memory. Linked memory consists of free and assigned areas.



**Fig. 3** Program segmentation gives the HP 3000 virtual memory. MPE automatically brings into main memory only those code segments that are currently needed. Thus a user's program may be much larger than main memory.

dex into code segment table or data segment table), disc address, priority, and frequency of access. This additional information is used in the selection of assigned areas to overlay when a request cannot be satisfied from the free area list.

### Virtual Memory

Virtual memory consists of main memory plus an area of mass storage called secondary memory, or the swapping area (Fig. 3). The swapping area is on disc or drum memory, although not necessarily on a single device; it may include areas of several devices. In the swapping area is a collection of pieces of code or data defined as segments. As a program executes, segments are swapped in and out of main memory by the operating system. Whether a segment is in main memory or absent, it is nevertheless part of virtual memory. Thus from the point of view of a user, he is working with a memory that appears to be many times larger than the actual physical size of main memory. His own program may exceed the 65K-word maximum main memory capacity and still allow space for many other users on the same machine.

As shown in Fig. 3, code is entered into the com-

puter in some source language, is translated to binary form by a compiler, and is stored in the file area. Each compiled program or subprogram exists in the file area as a relocatable binary module.

When the user is ready to execute his program, the appropriate command is given and the operating system loads the binary modules of his program into the swapping area of virtual memory. Simultaneously with this transfer, the binary modules are formed into segments as specified by the user. In some cases no actual change takes place; for example, a small program may consist of just one segment and the loader will probably not move it from a file disc onto the system disc unless the user wants this done.

Data segments are allocated dynamically when a program is loaded, and are always on the system disc.

### Scheduling/Dispatching

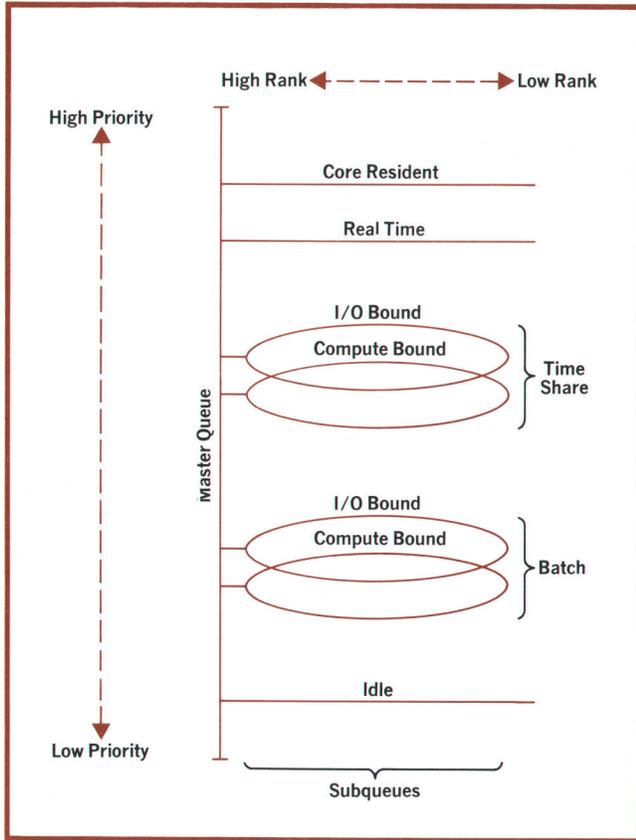
To accommodate the different modes of operation which may coexist under MPE/3000, the scheduling system is based upon a priority structure. All processes are logically organized into a linear master scheduling queue in order of their priority.

The dispatcher is responsible for allocating the central processor to the active processes in the scheduling queue. A process is considered active if it requires access only to the central processor. Otherwise, it is considered inactive, awaiting some other resource.

The basic organization of the scheduling queue is shown in Fig. 4. System processes are scheduled directly onto the master queue. Subqueues are used to schedule processes belonging to users. Note that since processes are scheduled independently, not all processes in a job are necessarily entered in the same subqueue.

There are five standard subqueues. Three are linear in structure. In a linear (sub)queue, the highest priority active process is given access to the central processor by the dispatcher, and it maintains this access until it becomes inactive or until it is preempted when a higher priority process becomes active. The three linear subqueues are for core-resident processes, real-time processes, and low-priority (idle) processes.

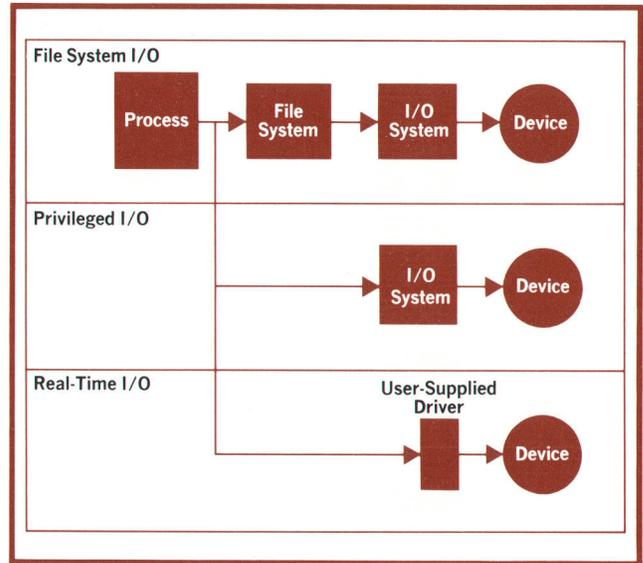
The other two subqueues are circular subqueues. These are for time-share processes and batch processes. In a circular subqueue, all processes are considered to be of equal priority and each active process accesses the central processor for a certain time interval. At the end of this time interval, the process releases the CPU and the next active process in the subqueue is dispatched. This continues in a



**Fig. 4** MPE schedules processes on the basis of priorities. Processes are organized into a linear master queue and five subqueues.

round-robin manner.

Each of the two circular subqueues is composed of two subqueues—a higher priority subqueue containing I/O-bound processes and a lower priority



**Fig. 5** Basic HP 3000 input/output access methods.

subqueue containing compute-bound processes. The dynamic rescheduling of processes between the dual subqueues is performed by MPE/3000. In the case of highly interactive time-share processes, this arrangement provides quicker response at the terminal.

### I/O System

The purpose of the MPE/3000 I/O system is to perform input/output operations for the file system. The user doesn't interact directly with the I/O system, but indirectly via the file system. However, privileged users may access the I/O system directly, and users with real-time capability may bypass both the file system and the I/O system for direct access to specific devices. Fig. 5 shows the basic I/O access methods.

In a typical I/O operation the sequence of operations is as follows. An executing user process generates a file request to the file system. The file system calls the attach-I/O intrinsic. Attach-I/O allocates an I/O queue entry and links it into the queue for the device specified. When all earlier requests for the device have been completed and the I/O monitor process has the highest priority among all other processes, the I/O monitor process begins execution of this request. There is one I/O monitor process for each device controller.

The I/O monitor process first assures that the data buffer is frozen in memory. The initiator section and the I/O program issue an SIO instruction to the device controller and return control to the I/O monitor process. Data is then transferred between the I/O device and the data buffer.

When the I/O monitor process is again dispatch-

## CHANGE OF ADDRESS NOTICE

The address shown is NOT correct.  
It should be as I have indicated below

\_\_\_\_\_  
\_\_\_\_\_  
\_\_\_\_\_

Remove my name from future JOURNAL mailings

Please complete the above and mail this section  
with address label on reverse side to:

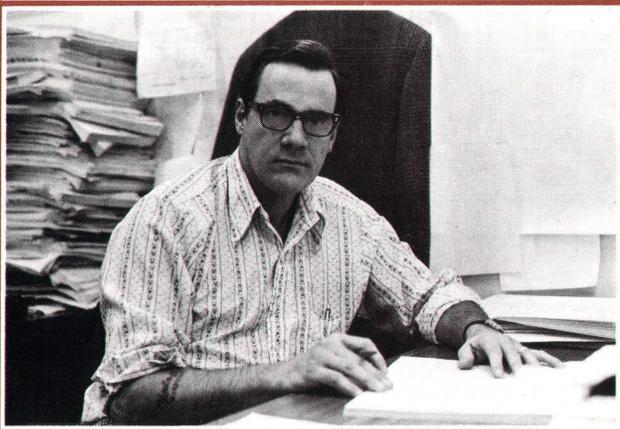
HEWLETT-PACKARD JOURNAL  
1501 Page Mill Road Palo Alto, California 94304

ed, it recognizes that an interrupt has occurred and calls the completion section of the device driver. The completion section checks for successful completion and returns the results of the I/O operation to the file system via the I/O control block. The user's process is activated upon I/O completion.

When the user process is again dispatched, return is made to the point following the file request.

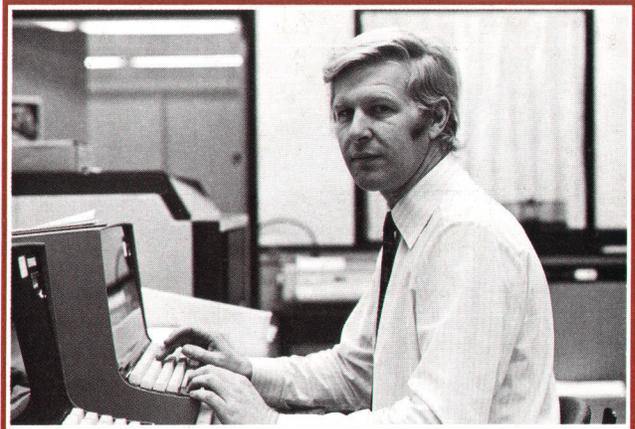
#### Acknowledgments

The following people were directly involved in the design and implementation of MPE/3000: Harlan Andrews, Larry Birenbaum, Terry Branthwaite, Jean-Michel Gabet, Jack MacDonald, Bob Miyakusu, Chris Larson, Tom Ellestead, Paul Rosenfeld, Steve Brown, and Myron Zeissler. 



#### Thomas A. Blease

Tom Blease's career in software design and implementation got its start in 1960 when he received his B.A. in mathematics from the University of California at Berkeley. In the ensuing years he held positions in that field with several organizations in Florida and California. At HP since 1969, he participated in the design and implementation of SPL and MPE for the HP 3000. He's a member of ACM and he enjoys a good hike on his days off.



#### Alan Hower

Alan Hower received his B.A. and M.A. degrees in mathematics from Christ's College, Cambridge University, England in 1960 and 1963, respectively. Between 1960 and 1970 he worked on software design and implementation with various companies in England and the United States. When he joined HP in 1970, he was first involved in the hardware design of the HP 3000. Later in the project he took on his recent responsibilities in the design and implementation of MPE/3000.

Address Correction Requested:

Hewlett-Packard Company, 1501 Page Mill Road, Palo Alto, California 94304

Bulk Rate  
U.S. Postage  
Paid  
Hewlett-Packard  
Company

## HEWLETT-PACKARD JOURNAL

JANUARY 1973 Volume 24 • Number 5

Technical Information from the Laboratories of  
Hewlett-Packard Company

Hewlett-Packard S.A., CH-1217 Meyrin 2  
Geneva, Switzerland  
Yokagawa-Hewlett-Packard Ltd., Shibuya-Ku  
Tokyo 151 Japan

Editorial Director • Howard L. Roberts  
Managing Editor • Richard P. Dolan  
Contributing Editors • Ross H. Snyder,  
Laurence D. Shergalis

Art Director, Photographer • Arvid A. Danielson  
Art Assistant • Erica R. Helstrom  
Administrative Services • Ruth G. Palmer