Data General

TECHNOLOGY REVIEW

32-Bit Superminicomputer Runs Both 16- and 32-Bit Code

The 32-bit ECLIPSE[R] MV/8000 is a compact superminicomputer based on VLSI technology that bridges the gap between 16- and 32-bit word programs to offer orderly transition from 16- to 32-bit computer systems. Announced by Data General Corp, Rt 9, Westboro, MA 01581 in response to increased application demands from users requiring performance and data handling power, the system features virtual address space of 4.3G bytes and user address spaces of up to 512M bytes each, built-in diagnostic processor, and hardware supported 8-ring security and protection mechanism.

Programmed array logic and VLSI technology reduce components allowing the the CPU to be implemented on five boards. The system consists of four major subsystems: CPU, memory, system control processor, and I/O system. The three separate processors supporting these subsystems—central processor, system control processor, and I/O processor—allow management of intelligence as close to the source as possible. This increases performance and minimizes data movement and system overhead. A series of high speed buses maximize processor throughput. These with the system's combination of instruction and system caches provide a memory system throughput rate of 36.4M bytes/s with I/O to memory and CPU to memory transfer rates of 18.2M bytes/s.

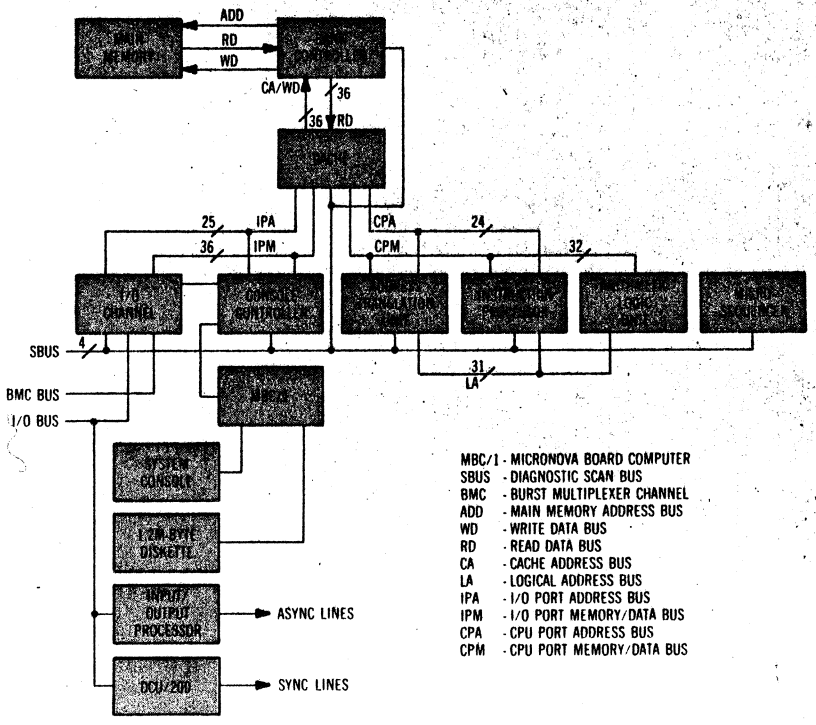
The CPU's 4-stage implementation allows instructions to be fetched, decoded, and executed simultaneously. A 1k-byte instruction cache in the CPU uses a lookahead lookback technique to increase speed. A RAM based microsequencer contains the memories that form the control store for microcode. Built around bit slices, the 32-bit ALU performs complex operations in a minimum of cycles. One section of the ALU operates on the exponents found in floating point numbers and the other works with floating point mantissas, fixed point quantities, and addresses.

Reliability and efficiency of the system's memory are enhanced by incorporation of an error detection and correction mechanism called "sniffing." This technique consists of the bank controller reading one double word from each RAM row as it is refreshed or recharged. (The bank controller is the interface between system cache and memory modules.) The double word goes through a complete error detection and correction check and is returned to the memory module. Because the operation is performed on a different double word during each refresh, the entire contents of memory are checked and corrected if necessary every 4 s.

Memory modules are available in increments of 256k bytes up to 2M bytes maximum, and are organized as 64k double words of 4 bytes each. Seven additional bits stored with each double word provide error checking and correction facilities; 4-way interleaving offers maximum performance. Data transfer between memory modules and system cache at a 36.4M-byte/s rate is possible because of overlapping of memory operations.

The 16k-byte system cache acts as a high speed buffer between memory modules and the rest of the system, reducing the time needed by both CPU and I/O system to access memory.

(continued on page 48b)



Data General's 32-bit Eclipse MV/8000 computer system supplies 512M characters of user program address space and main memory bandwidth of 36.4M characters/s. Machine uses a microprogrammed processing unit, block oriented memory system with 16k-byte cache, and independent I/O system, and provides integrated microcomputer based diagnostic capability for comprehensive system control and error diagnosis

Interface between system cache and main memory transfers data at a rate of 16 bytes in 550 ns for write and 16 bytes in 440 ns for read operations.

A hardware accelerator for the memory paging system, the address translation unit maintains a table of up to 256 address translations and access privileges for recently referenced pages. The unit also generates modified and referenced bits. Modified bits signal when a page in main memory must be written back to disc because it has been modified; referenced bits indicate that the page was referenced and are used by the operating system to perform page replacement algorithms.

Made up of a high speed burst multiplexer channel, data channel, and I/O processor, the I/O system is under the control of the I/O channel board. All data transferred within this system move to or from system cache; they need not pass through or interfere with the processor. Transferring blocks of data at up to 16.16M bytes/s the BMC is a direct communications path between main memory and high speed peripherals. The data channel handles transfers between CPU and medium speed devices operating at rates up to 2.27M bytes/s. An independent processor with ECLIPSE instruction set and 64k bytes of local memory, the IOP processes and buffers data from low speed units. Acting as a frontend processor for the system, it offloads the CPU and receives terminal output from the CPU.

Designed in conjunction with the computer, the concurrently announced Advanced Operation System/Virtual Storage (AOS/VS) takes advantage of hardware architecture to manage system resources for 128 users accessing up to 512M bytes of logical address space. The software also supports simultaneous execution of 16- and 32-bit programs and can run 16-bit programs developed under AOS without modifications and with increased speed. Support software for use under AOS/VS includes three 32-bit languages that conform to ANSI standards: FORTRAN 77, PL/I, and BASIC. In addition, the operating system offers a 32-bit SWAT™ Native Language Debugger.

The intelligent multiprogramming system controls concurrent timesharing, batch, and online operations for both 16- and 32-bit processes. Working dynamically with hardware, the system operates on a demand paging basis using pages of 2k bytes. It uses a cache based hardware address translation unit to accelerate the translation of logical addresses into virtual addresses.

Segmentation of virtual memory into eight processing regions provides the hardware for an 8-ring software security mechanism. The structure permits the operating system to reside in the user's address space while protecting it from user encroachment.

Compilers include a common code generator and optimizer to select the most efficient code sequence. Common language modules are used, further increasing system reliability and maintainability.

Representative prices for the systems range from $153,150 to $504,700. On the low end the system consists of CPU with 512k-bytes memory, battery backup, system console, 8-line asynchronous modem interface, 96M-byte disc, and 800/1600-bit/ in magnetic tape. The high end system has 2M-bytes memory, and adds three 16-line asynchronous terminal interfaces, 64 CRT displays, four 277M-byte discs, magnetic tape unit, and 900-line/min printer. Deliveries are scheduled for October.

Circle 420 on Inquiry Card

Information Processing Systems Support Multiple Hosts, Multiple Protocols

Initially available in two models, the 9200 family of microprocessor based information processing systems allows for advanced networking capabilities and modular addition of functions. Both local and remote communications in either SNA/SDLC or BSC networks are supported by the units from Harris Corp, Data Communications Div, 16001 Dallas Pkwy, Dallas, TX 75240, to enable users to change protocols with minimum effort.

Ability of the system to support concurrent communications with multiple hosts will provide a major advantage that will enable users to combine operations under one interactive system for efficient utilization of communications lines and equipment. A user with host processors in two separate cities operating in BSC or SNA will be able to communicate with the 9200, which will handle communications with each host concurrently.

Local attachment to IBM host systems is achieved via byte, block, or selector channel in 3272 or SNA mode. Remote connection uses either BSC or SDLC protocol. Speeds up to 9600 bits/s are supported by 9200 processors.

When operating in BSC protocols the 9200 will communicate over duplex or half-duplex facilities in ASCII or EBCDIC. When in the SNA/SDLC environment, the unit will be capable of local channel attachment or remote communications in half-duplex, flipflop send/receive modes.

The basic 9210 will support up to 32 devices per system with local attachment at channel speed, and remote communications up to 9600 bits/s. When upgraded to the 9220 model, this system will also support more than one host concurrently.

Systems are tailored to individual requirements by insertion of diskettes containing parameter definitions. This allows users to specify printer authorization, screen configuration, and number of devices, and to reconfigure the system by entering new system parameters.

Among the system peripherals are a 15" (38-cm) nonglare CRT; keyboard configurations including 75- and 87-key data entry typewriter, and keypunch styles; and printers that include both dot matrix bidirectional and band printers. Other options include a photopen light sensor and magnetic slot reader.

A basic 9200 system includes processor, six display stations, and one 130-char/s bidirectional printer. Purchase price is $23,306.

Circle 421 on Inquiry Card

(continued on page 48f)

32-BIT MINICOMPUTER ACHIEVES FULL 16-BIT COMPATIBILITY

Early architectural definition results in design of 32-bit minicomputer that offers full 16-bit compatibility without a mode bit

Steve Wallach
Chuck Holland

Data General
Route 9, Westboro, MA 01581

Designing a viable product for the competitive 32-bit minicomputer market requires a specialized approach that combines the latest semiconductor technology with a system organization capable of achieving the best possible performance. When the 32-bit minicomputer must offer full 16-bit compatibility without enforcing two distinct modes of operation, the specialized design approach becomes even more important. Conventional design techniques constrained in this way may lead to a tradeoff between performance and ease of implementation. However, by having implementers and architects interact in a top-down, goal driven design effort at the earliest phase of development, compromise of performance is avoided.

Undertaken over a 2-year period, the design of the ECLIPSE MV/8000™ involved cooperative implementation and architectural development. Critical elements of that process included the structure and functions of the design team; the basis for design decisions; an overview of the target system organization, particularly its instruction set; and the significance of specific architectural features. Important factors to be considered were the reasons for designing a 32-bit machine and the reasons for avoiding two complementary modes of operation.

As declining component costs enlarge the 32-bit technology market, other applications are developed and demand for these application systems steadily grows. The major advantage of the 32-bit machine is its ability to support a large logical address space for storing programs and data. Because of this, the design had to provide the largest possible address space constrained to 32-bit words—a 4G-byte size—and to allow use of the complete address space in the first implementation. Important to this part of the design was the concept of equating files with the address space, first developed under MULTICS[2]. Finally, because structure and organization of the address space strongly influence

system capability in a multiprogramming, multi-user environment, a major portion of the design effort dealt with the layout and management of the 4G-byte address space.

Compatibility through use of a mode bit is predicated on two mutually exclusive instruction sets with the constraint that only one instruction type can function at a time. Also, marketing considerations indicated the need for a 32-bit minicomputer that featured total compatibility with existing 16-bit software. Six years ago, instruction set extensions in the 16-bit ECLIPSE® computer achieved full binary compatibility with the 16-bit NOVA® computer, introduced more than 12 years before that, without resorting to the use of a mode bit. It therefore seemed prudent to use the same technique for extending to 32 bits. In addition, without a mode bit the 16-bit instruction set is not detachable, thus ensuring a continued commitment to software compatibility. This nondetachable instruction set results in binary compatibility with Advanced Operating System (AOS) programs based on the 16-bit ECLIPSE and allows 16-bit program development on the 32-bit ECLIPSE MV/8000 system. It also supports concurrent execution of 16- and 32-bit programs and permits mixing 16- and 32-bit subroutines (or individual instructions) within the same program.

To best accomplish the design of a 32-bit machine without a mode bit, the project team was divided into hardware and software implementation groups. Establishment of a cohesive team was just as important to the success of the project as technical judgments made during the design effort. Partitioning the design team into two groups instead of designating a committee of architects expedited definition of the overall hardware and software implementation. Since, ultimately, system implementers would have to conform to the overall hardware and software definition, design team members felt that intensive interaction at the earliest phases of the design effort was a prerequisite for implementation of the architecture. As will be shown, many architectural features of the instruction set were in fact influenced by the first implementation.

Suggestions for architectural enhancements and alternatives were evaluated as to their impact on hardware and software development schedules, the hardware resources needed to implement proposed features, their effect on performance, and perceived marketing needs. Although responsibility for final acceptance rested with the appropriate hardware or software team leader, suggestions were welcomed from all implementation team members and other technical personnel within the central engineering and software development groups.

Hardware Organization

Before final definition of the architecture, logic designers and microprogrammers on the hardware implementation team developed an overall system organization (Fig 1) based on a dual-port cache memory system, with the processor divided into four basic components handling input/output (I/O), main memory access, central processing, and system control (operator console, clock generation, etc). This system organization was chosen after a detailed system analysis had been done taking many factors into consideration. For example, handling of I/O data transfers involves block oriented (eg, disc based) operations that access main memory directly and require appropriate logic for detecting that a copy of the designated physical block resides in cache memory. It also involves single word transfers that require either buffering or partial read and write accesses to main memory. I/O buses transfer data in 16-bit words. The memory system in a 32-bit machine usually grants requests in integer multiples of 32 bits. Therefore, system organization had to make some provision for accessing data of less than 32-bit precision.

System organization also had to support extremely high bandwidth processor and I/O subsystems, both able to exceed 16M bytes/s. Furthermore, it was important to minimize the time for cache miss (cache fault) resolution. Most cache designs are optimized to give the highest possible hit rate (cache hits/attempts, or 1 – fault rate). While this is always desirable, the true measure of cache efficiency is the effective access time:

(hit rate) (cache access time)
+ (fault rate) (main memory access time)

Therefore, the interface between the cache and main memory had to be optimized to reduce the overhead associated with a cache fault. Ultimately, this goal necessitated the construction of a dual-port cache featuring an I/O port which exhibited characteristics of a general purpose 32-bit bus, and a central processor port. Other factors influencing system organization included a desire to minimize the interface between the I/O subsystem and the processor subsystem, the need to support a soft operator's console based on a video display, and the inclusion of sophisticated diagnostic subsystems.

Microsequencer Features Random Access Memory Control Store

Based on a random access memory (RAM) control store, the microsequencer contains the microcode that interprets ECLIPSE MV/8000 instructions. The microcode, which is organized as 4k by 75 bits for a 74-bit microinstruction with parity, generates control signals required by the other processor elements. Because the control store was designed using RAM, the microcode is reloaded from a 1.2M-byte diskette by the system control processor (SCP) each time the system is powered up.

The use of RAM achieves control store alterability, making microcode changes or updates easy to implement by simply replacing the diskette, and providing microdiagnostics. When a malfunction is suspected, microdiagnostics overlay the control store to give accurate fault isolation in a matter of minutes at a finer granularity than can be achieved using diagnostics based on standard machine instructions.

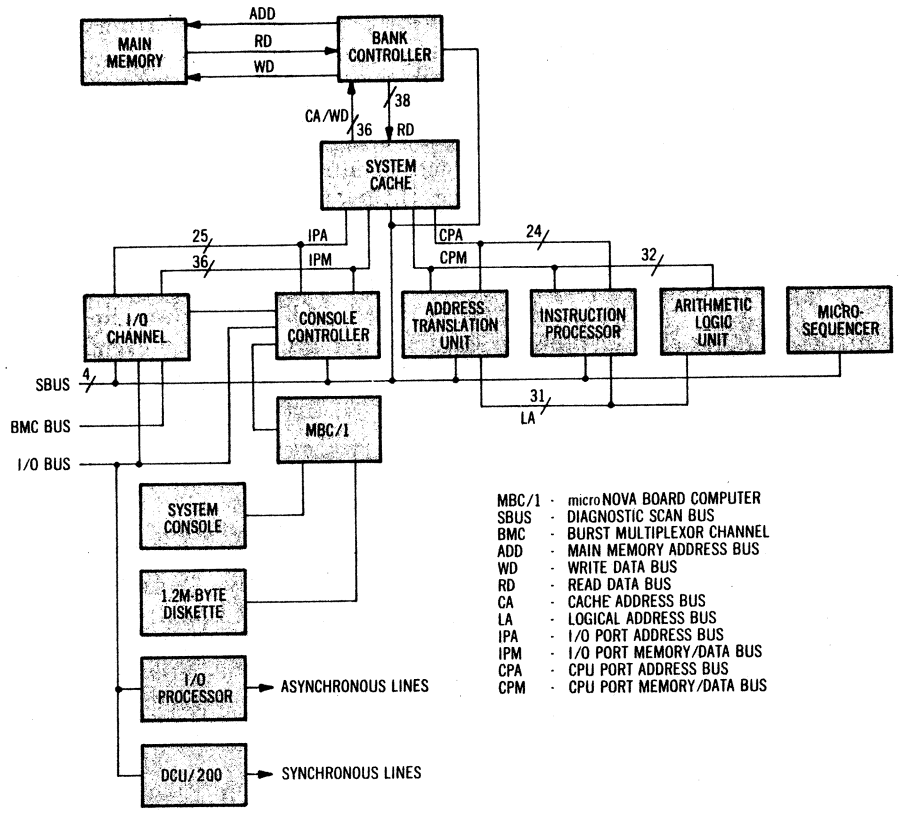


Fig 1 System block diagram. Separated here by function, processing elements interconnect in pipelined, asymmetric multiprocessor built on five standard 15 x 15" (38 x 38-cm) circuit boards. With 110-ns minor and 220-ns major cycle times, instructions typically execute within 440 ns

Programmable Array Logic Functionality Reduces Part Count

More than any other component type, programmable array logic (PAL) influenced the design, performance, and personality of the ECLIPSE MV/8000 system. This family of devices implements a programmable AND array that provides input to a fixed NOR array and has registered (D flipflop) and combinatorial versions packaged in 20-pin, 0.3" (0.8-cm), dual-inline packages. The use of PAL components began slowly. When designers started achieving a three- to fivefold reduction in parts count over the use of off the shelf, medium scale integration (MSI) components, PAL usage accelerated. Ultimately, more than 10% of all central processor parts were PAL components, distributed about 40, 30, and 30% among the 16R4, 16R8, and 16L8 part types, respectively. The ability to feed back registered PAL outputs internally motivated the high usage of 16R4 and 16R8 part types.

For the ECLIPSE MV/8000 computer, PAL proved to be the right idea at the time. Its high density package combines a high level of functionality with the potential for reduced parts count; hence, design modifications require only fuse changes and not changes to the printed circuit board etch. Additionally, PAL components are easy to use. Another interesting aspect of PAL emerges from an examination of the cases in which it was not used, cases when an off the shelf MSI design could be achieved at a lower cost, or cases when a PAL implementation would have been too slow. For example, a 16L8 has a propagation delay of 40 ns. In many cases, this was not sufficient to handle the worst case delay through the network. If PAL components with a propagation delay of 20 ns had been available, their use would have increased significantly.

Pipelined Instruction Decoding Improves Performance

Unique to the 32-bit minicomputer industry, the instruction processor, which decodes instructions for subsequent execution, contains a 1k-byte direct mapped instruction cache organized as a 64-block, high speed memory of 16 bytes/block. Because of its look ahead and look behind potential, the instruction cache speeds up program execution, particularly benefiting program loops and backward branches. Totally separate from the system cache, the instruction cache allows instructions and data to be fetched concurrently without interference.

Instruction cache output provides input to the instruction decoder. A high degree of pipelining allows instruction execution to proceed through four distinct stages. An instruction is fetched from the instruction cache, the instruction operation code is decoded to obtain the microcode starting address, the first

micro-instruction is read from this address, and finally, the first microinstruction is executed. Operating at 110 ns/stage, the 4-stage pipeline has the ability to fetch and completely decode the next instruction while the present instruction is being executed, overlapping the first two stages of the subsequent instruction with the last two stages of the current instruction. Consequently, the overhead normally associated with instruction decoding is almost entirely overlapped with instruction execution. This results in a significant performance increase when compared with processors that lack the feature.

SCP Targets on Availability, Reliability, and Maintenance

The SCP, a microNOVA™ controlled diagnostic and console monitor, is the control element of the ECLIPSE MV/8000 system availability, reliability, and maintainability (ARM) mechanism. A console control board provides all system timing. In addition, the microNOVA board computer (MBC) to system bus (SBUS) interface originates on the console control board, which also contains the realtime clock and the programmable interval timer. Driven by a universal asynchronous receiver/transmitter (UART), the diagnostic SBUS connects all major subsystems: the address translation unit (ATU), microsequencer, system cache, memory bank controller, instruction processor, I/O channel, console control, power supplies, and expansion cabinetry. And, as mentioned earlier, microdiagnostics isolates faults to a particular hardware board in a matter of minutes.

SCP power margining, used to stress logic components and isolate components that may be failing intermittently, is accomplished by increasing power by as much as 5% above normal or decreasing it by up to 10% of normal voltage. System clock margining serves three functions. By increasing the system clock to 120 ns, it permits the processor to operate reliably when logic boards are on board extenders and allows a component to function when it has degraded beyond performance specifications, a feature used by the diagnostics. By decreasing the system clock to 100 ns, it stresses logic components and helps to isolate intermittent component failures.

The SCP also regulates the ability to disconnect the instruction cache, system cache, and ATU address cache, allowing continued processing in a degraded mode of operation. In a modern, highly interactive, online application environment, ARM is as important as any of the critical system parameters, such as price or performance. To be effective, it must be tightly integrated into the design, not added at the last minute. Overall, the SCP control, diagnostic, and margining capabilities provide the required level of effectiveness and functionality to speed up maintenance and repair and minimize reasons for shutting down the system.

Sniffing Protects Memory Data

A sniffing mechanism reduces the likelihood of encountering uncorrectable multiple bit memory errors, thus greatly increasing data reliability. All memory boards are refreshed simultaneously. Each time a memory refresh operation begins, the bank controller reads one 16-byte memory block from one of the memory boards, subjects the block to a complete error check and correct procedure, and writes it back to the memory module. Because sniffing for errors occurs on a different block during each refresh operation, the entire content of main memory is checked and corrected at the rate of about 1M bytes/s. Sniffing prevents accumulation of single-bit memory errors that might become uncorrectable double-bit errors. By guaranteeing that every main memory location is checked and, if necessary, corrected within a known time period, it prevents errors from accumulating even in memory locations that are not referenced by a program.

Address Space Structure

Because the ECLIPSE MV/8000 computer is in fact a 32-bit ECLIPSE computer, development of a preliminary instruction set proceded quickly. Once this phase was complete, attention was focused on defining the logical address space because further instruction set evolution hinged on its structure. From the outset, three high level objectives were the basis for evaluating all proposals. The logical address space had to be easy for the operating system to manage. Larger applications involving more than 1M bytes of memory required an efficient translation mechanism that did not compromise 16-bit applications or smaller applications involving less than 1M bytes of memory. Sessions with customers whose needs surpassed the capabilities of a 16-bit machine revealed that relatively small programs in the 250k- to 500k-byte range were projected, especially in commonplace realtime environments. Finally, the operating system had to be embedded into the logical address space so that both systems and applications software would benefit from the address space extension.

When establishing memory management, the first step taken was choice of page size. A 2k-byte page was selected on the basis of the average file size under AOS, the page fault rates achieved by various other page sizes, the overhead incurred during page frame management, the working set size of a fixed entry ATU, and the effect of page size on the logical to physical translation mechanism. [3,4] (Fig 2 compares the page table overhead for 512- and 2k-byte page sizes.) This choice is an important one because the 2k-byte page also becomes the basic unit of memory protection. As will be shown, another consequence of this choice is that 2k bytes becomes the largest unit of contiguous physical memory under every logical address space size, which simplifies operating system design and requires maintenance of only one free page list. The selected page size immediately determines the 11 least significant bits in a 32-bit logical address, leaving the semantics of the remaining 21 bits undefined. The two fundamentally different approaches to interpreting these 21 bits were christened "flatlands" and "treelands" by the design team.[5]

The flatlands approach involved hashing the 21 high order bits of a logical address into a translation region

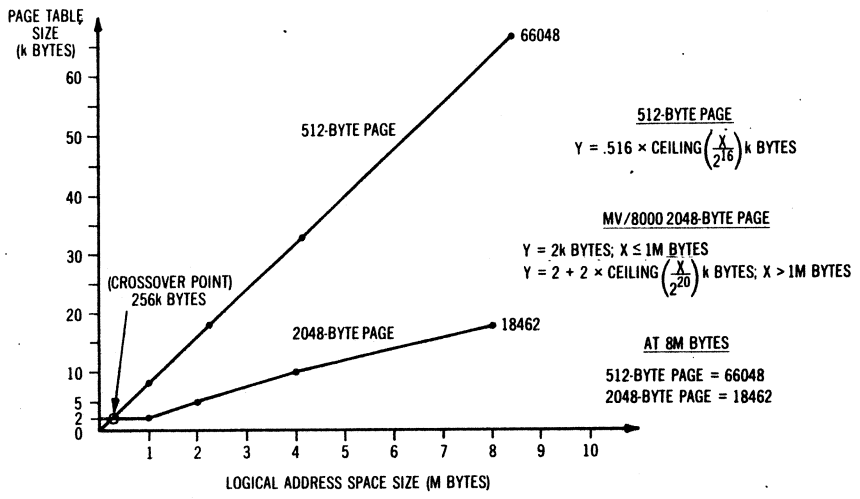


Fig 2 Page table overhead. Plot of overhead created during page frame management compares impact on performance of two candidate page sizes during logical to physical address translation

within physical memory. Acceptable performance required open addressing, linear probing with a 50% load factor on the bucket size. This would have resulted in an average of 1.5 probes for a cache hit and an average of 2.5 probes for a cache fault.[6]

The treelands technique, the one ultimately implemented, used a more traditional page table approach. In this form of logical address translation, a table constructed in main memory maps logical addresses into physical addresses. Each page table entry supplies various flags and, if appropriate, the physical address that corresponds to a logical address. With the selected page size of 2k bytes, it was assumed that four bytes would be sufficient for physical address information, and the page table, therefore, would have 512 entries.

Performance was the determining factor behind the choice of a page table approach. In view of the expected page fault rate, the two memory references needed to resolve address translation using the hashed technique, and the performance characteristics of the processor microarchitecture, the page table exhibited better performance than a hashed index. Had the logical address required a 3-level lookup, the hashed approach would have been more efficient.

Because of its structure, the page table requires a 9-bit index. Two such indexes are used to construct 18 of the 21 logical address bits that remain unresolved after taking the page offset into consideration. Since use of a third, 3-bit page index field was considered wasteful, the remaining three bits reference one of eight segment registers maintained within the processor.

The treelands approach was further optimized in two ways. One permits a 1-level translation for those programs that use less than 1M bytes on a segment basis, with a bit in the segment base register denoting either a 1- or a 2-level translation. The other adds validity bit interpretation in each page table entry for segment boundary protection against out of bounds references. Because a page is allocated to each page table, it is not necessary to define segment length registers. Instead, AOS/VS, the 32-bit operating system that manages MV/8000 resources, simply marks as invalid any page table entry that maps an undefined portion of the logical address space.

Studies by Belady and Hatfield show that complete support of referenced bits is important to the efficient management of a large virtual address space.[7,8] Three alternative locations for these bits were considered: the page table entry associated with the page in main memory, the part of the machine state attached to each page frame of the physical address space for the implementation, and a table associated with the process currently running. Each alternative was analyzed in relation to the time needed for software to access the referenced bits (ie, the work factor associated with the page replacement algorithms), the impact on the present and future implementations, and the effect on performance and design of the system data and address cache accelerators. Associating referenced and modified bits with the machine state, similar to the method used in the IBM 370, proved best.[9] The availability of high density static RAM and the ease of defining special instructions to accelerate page replacement were major determining factors, as well as the simplification of the ATU, which then need only be read with respect to main memory.

Privileged instructions were next defined to manipulate referenced and modified bits. One group of instructions permits each referenced and modified bit of a page frame to be examined individually. The other allows referenced bits to be treated as a bit string. The OR Referenced Bit (ORRB) instruction can perform the inclusive OR of a string of referenced bits and a string in main memory, resetting the accessed bits to zero. This approach helps the operating system maintain referenced bit strings easily on a per process basis, using the standard set of bit string instructions (count bits, locate first bit, etc) for string analysis. Various

algorithms maintain multiple bit strings per process to simulate reference counts greater than one.

Address Space Protection

Because the operating system is embedded in the user's context, some form of protection must be provided to detect and prevent malicious or accidental encroachment into the system address space. The domain model and the capabilities model were analyzed, but capabilities were eliminated during the initial phase because the complexity of the added hardware was not commensurate with the return.[10] It was not desirable to restrict capabilities to special segments, to aid the granting and revocation mechanisms, nor was it desirable to expand the logical address beyond 32 bits or append tags to each memory location with capability potential.[11] This is not to say that capabilities were undesirable in their own right, but that the proper support mechanism seemed to be beyond the scope and objectives of the project.

The domain model was narrowed to a hierarchical approach using one of the possible ring implementations. Since either a logical address or an integer can occupy an accumulator, and most instructions do not distinguish between integers and addresses, the MULTICS model could not be used directly. Two alternatives were considered: binding the ring number to the address space, and defining a ring register to be maintained separately. The first abstraction was chosen to eliminate the need for an extra machine state requiring protection, to permit the same instructions used for subroutine calls and returns to be used for cross-ring operations, and to add to the efficiency of Trojan horse pointer detection.

With the decision to bind the ring number in the address space, the number of rings had to be specified. In its early design phases, the operating system was modeled to use four rings, with an additional four left available for application software (Fig 3). An evaluation made to determine whether this static binding would overly compromise the general structure of the logical address space revealed that, from an operating system viewpoint, the static binding would not interfere and that if access brackets were implemented, they would in fact generally force this binding. Moreover, large user programs exceeding 1G bytes would use the lower numbered rings for program and stack storage and the higher numbered rings for static data.

The inclusion of read, write, and execute protection on a per page basis and a gate structure supported by hardware completed the protection system. Coupled with a trap mechanism, which is always precise because it maintains the address of the offending instruction, these protection mechanisms simplify the debugging cycle and generally abort runaway programs instantly. The gate array, defined for each ring, permits a called routine to designate the number of gates and an access bracket for each gate during all inner ring calls (Fig 4). Valid cross-ring calls, complete with stack switching and construction of a new frame on the target stack, take about 6.6 μs, without requiring software intervention. This level of effort directly supports the objective of processing operating system calls quickly and efficiently.

Instruction Set

Standard 16-bit ECLIPSE and NOVA computers, instructions form the basis for the ECLIPSE MV/8000 instruction

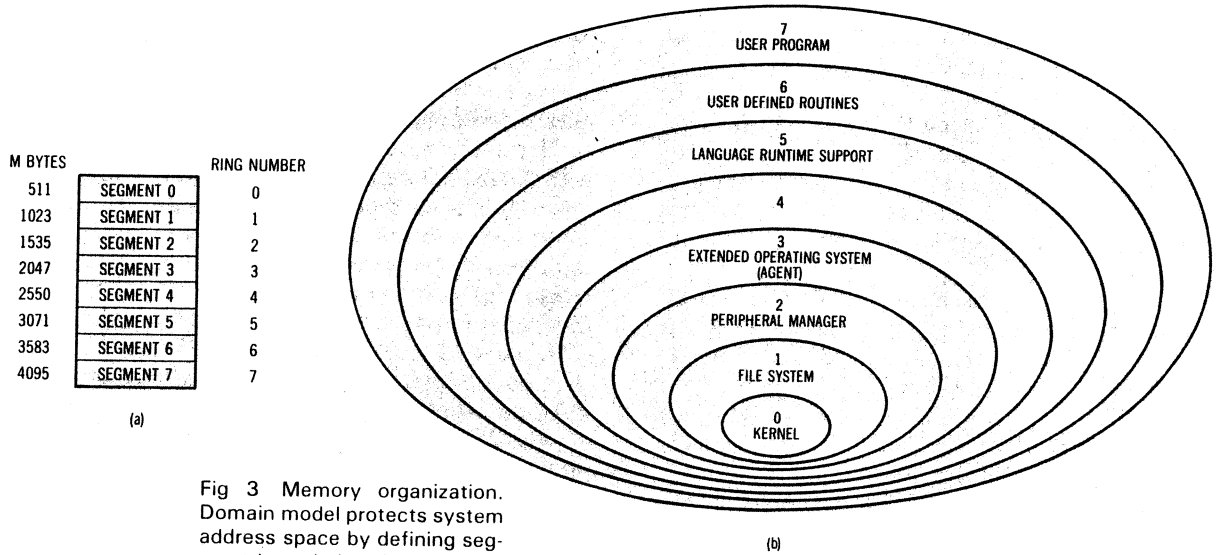


Fig 3 Memory organization. Domain model protects system address space by defining segment boundaries shown in (a). Then, in (b), AOS/VS hierarchical software system structure assigns memory segment and category or layer of software to each ring

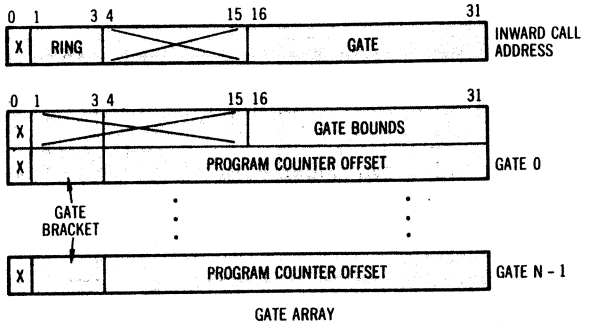


Fig 4 Gate mechanism. Called routine designates number of gates and access bracket for each gate during inner ring calls. Valid intra-ring calls execute in 6.6 μs, including stack management overhead. Both processes enhance operating system performance

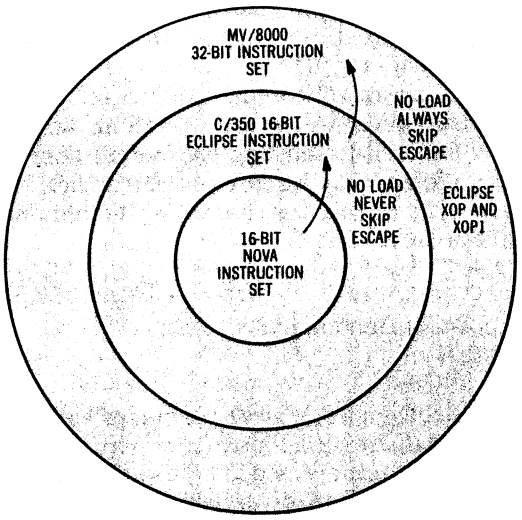


Fig 5 Hierarchical instruction set relationships. To achieve hierarchical mode-free instruction set, MV/8000 extends ECLIPSE instruction set to handle 32-bit accumulators and 4G-byte address space. New instructions use ECLIPSE no load, always skip, escape category in the same way ECLIPSE instruction extended NOVA no load, never skip, escape category

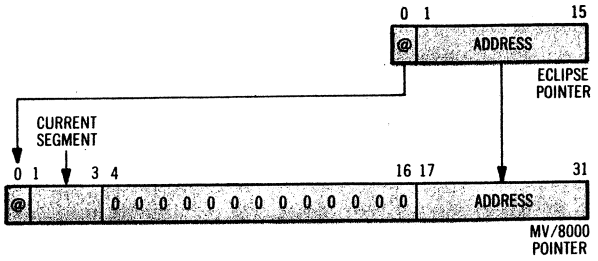


Fig 6 Pointer relationships. To dynamically map 16-bit address space into 32-bit address space without loss of performance, all 16-bit addresses are constrained to reference the first 64k bytes in segment

set. Extensions to manipulate the 32-bit address space and the 32-bit fixed point accumulators, and to expand general purpose functionality, are achieved by defining a new instruction class using the no load, always skip, binary encoding of the ECLIPSE ALC instructions (Fig 5). Four 16-bit accumulators are expanded to 32 bits, with their least significant 16 bits serving for 16-bit operations. Interpretation of fixed point data as 2's complement numbers accounts for this arrangement. Expanded to 31 bits, the program counter (PC) references memory at a 16-bit word granularity. The four 64-bit floating point accumulators (FPACs) are identical to those in the ECLIPSE computer. FPACs contain only floating point data—never addresses data—hence no reinterpretation is needed. Expanded from 32 to 64 bits, the floating point status register contains the 31-bit floating point program counter, which is a copy of the PC when a floating point fault occurs. This ensures that floating point faults are always precise and independent of the pipeline or overlapped nature of floating point computations. Two additional machine state structures enhance efficiency and fault handling capability: a 16-bit processor status register and four 32-bit stack management registers.

In the 16-bit ECLIPSE computer, a stack pointer, stack frame pointer, and stack limit maintained in hardware assigned memory locations manage the stack. Four 32-bit registers manage the ECLIPSE MV/8000 stack: a stack base register for detection of stack underflow, a stack limit register to detect overflow, a stack pointer, and a frame pointer. These parameters migrated into registers for performance reasons. The 16-bit processor status register (PSR) contains the flags used for integer overflow interpretation. One flag, OVR, signals that overflow has occurred. The other flag, OVK, is used as a mask to initiate or suppress the integer trap mechanism. Remaining bits are reserved for future hardware augmentation.

Following preliminary instruction set definition came the design of the logical address space structure; once that was completed, instruction set definition resumed. However, one final attribute of compatibility remained unresolved: mapping the 16-bit address space into the 32-bit address space. This was achieved by constraining a 16-bit address to the first 64k bytes of each segment (Fig 6), which permits efficient AOS/VS management of 16-bit processes running concurrently with 32-bit processes and allows instructions that generate 16-bit addresses to be interspersed with instructions that generate 32-bit addresses in the same program.

Language Viewpoint Attains Symmetry

Language related and system related constraints divided the instruction set definition process into two areas that were by no means mutually exclusive but are best discussed separately because each has different objectives. From a high level language viewpoint, the objective was to make the instruction set symmetrical with respect to all relevant data types. Consequently, all memory to accumulator and accumulator to accumulator instruction types apply to the baseline 16-and

32-bit integer data types and the baseline 32- and 64-bit floating point data types. Additional instructions provide a rich repertory of immediate operations, comparisons, shifts, memory to memory increments and decrements, logical and Boolean bit strings, byte and byte string manipulations, and sophisticated transfer of control (including CASE and DO WHILE constructs). Addressing uses the standard ECLIPSE computer modes: absolute, PC relative, and indexed through an accumulator.

System Viewpoint Adds Capability

From an operating system viewpoint, the most important instruction attributes are not the instructions per se but the capabilities of individual memory reference instructions to manipulate user data bases directly in a secure, efficient, and reliable manner. Once these attributes were firmly established, effort concentrated on providing instructions that could replace sequences of repetitive operations. The first step was to identify an instruction set to properly manipulate machine state constructs defined for the 32-bit logical address space. This involved the segment base registers (SBRs) and the referenced bits and modified bits maintained by the hardware.

Approached from an operating system perspective, the 4G-byte logical address space is divided into two regions: a system wide 512M-byte kernel and a process or user wide 3.5G-byte region (Fig 7). Directly supporting this abstraction, the Load All SBR (LSBRA) instruction purges the entire ATU, and the Load Some SBR (LSBRS) instruction effectively purges the ATU of entries associated with segments 1 to 7, inclusive. ATU purging is needed to avoid associating the user logical address of the currently executing process with the translation context of a previously executing process.

Another class of instructions deals specifically with the referenced and modified bits. Many iterations, some of which followed performance measurements on an engineering prototype, led finally to the concept of treating referenced bits as a group of 16-bit (rather than binary) strings. The 16-bit string structure reduces the page replacement overhead by about 70%. It is noteworthy that without the alterable RAM based microsequencer changes of this magnitude could not have been accomplished so late in the development cycle.

Once AOS/VS coding was under way, operating system designers requested a comprehensive set of queue handling instructions. Their request was based on efficiencies in code density; performance in the scheduling and resource management modules; and reduced interrupt latency, an important realtime parameter that measures the delay in honoring an interrupt request. AOS/VS uses a priority queue model. It has only one physical queue; however, areas within the queue are grouped by priority level (Fig 8). Therefore, insertions are necessary not at the beginning or the end of the queue, but only relative to a specified queue element.

Two types of queue instructions were defined to meet efficiency and performance goals. One permits queue insertions and deletions, and the other permits searching the entire queue under particular matching criteria. Various combinations of direction (forward or backward), data type (16- or 32-bit) and match relation (bitwise or unsigned comparison) required 32 queue search instructions. All queue operations were defined to allow linked list modeling and permit noninterruptible operations or, in the case of the searches, to return unique results when a search must be discontinued temporarily because of an interrupt.

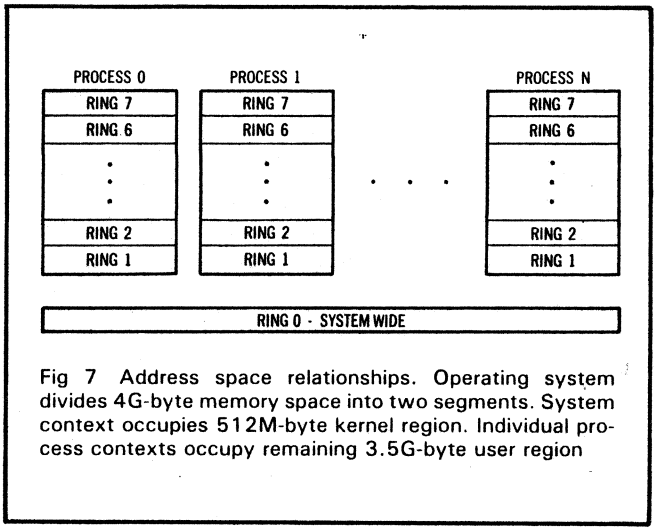


Fig 7 Address space relationships. Operating system divides 4G-byte memory space into two segments. System context occupies 512M-byte kernel region. Individual process contexts occupy remaining 3.5G-byte user region

Breakpointing Demonstrates Design Interaction

Program debugging is a great portion of software cost. Therefore, as part of the instruction set definition process, the design team provided hardware support for the existing SWAT™ high level language debugger, one which assumes that a breakpoint instruction can replace the first 16 bits of any other instruction. Breakpoint

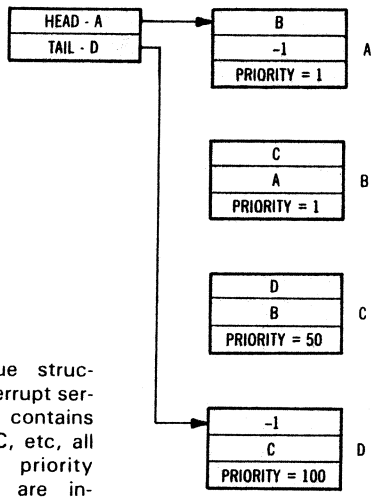


Fig 8 Queue structure. One interrupt service queue contains areas A, B, C, etc, all grouped by priority level. Items are inserted into appropriate area; therefore, they need not enter queue at beginning or end. Instead, special instructions position new items relative to queued items and also search queue under match criteria

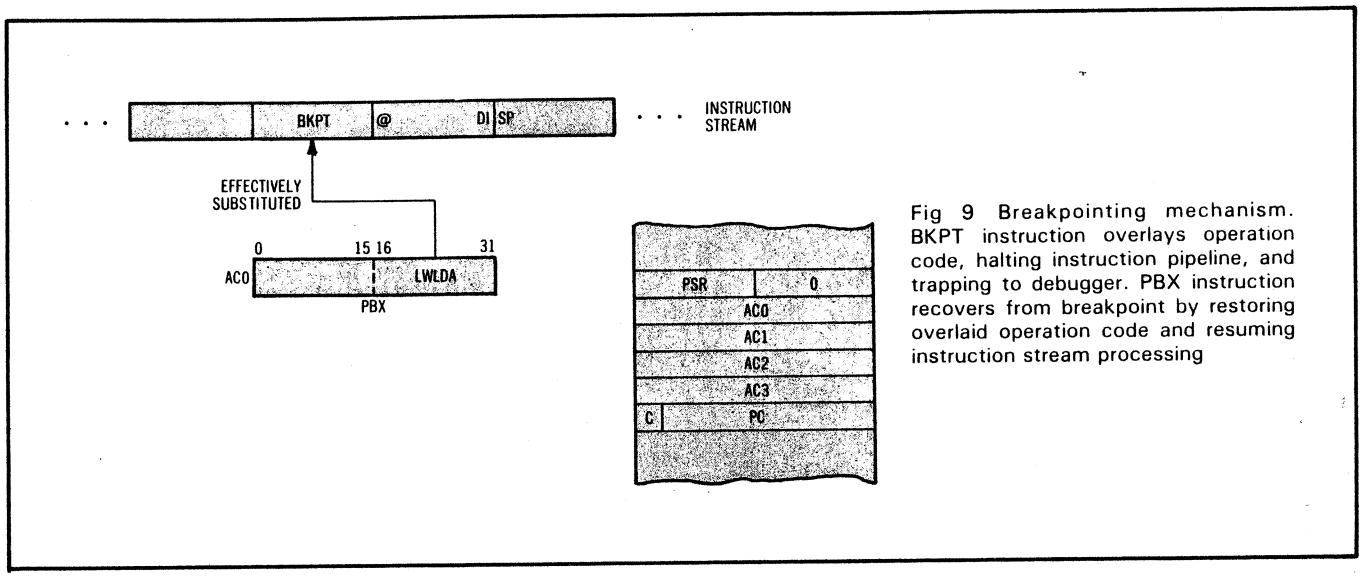


Fig 9 Breakpointing mechanism. BKPT instruction overlays operation code, halting instruction pipeline, and trapping to debugger. PBX instruction recovers from breakpoint by restoring overlaid operation code and resuming instruction stream processing

(BKPT) instructions are constrained to be exactly 16 bits long. Moreover, they must provide a trap address referencing the first instruction of the trap handler. In keeping with the overall system trap concept, a page 0 trap handler location was defined within the first 512 bytes of each segment so that each process and/or ring could have its own debugger.

Complementing the BKPT instruction, the Pop Block and Execute (PBX) instruction effectively restores the breakpointed instruction opcode, substituting it for the BKPT opcode (Fig 9). Superficially, this instruction pairing seems straightforward and easy to implement. However, during the prototype debugging phase, deleterious side effects of the initial instruction definition and implementation became apparent. One side effect was the potential to return to a deleted breakpoint: when a BKPT instruction was removed and replaced with the actual opcode, the debugger still returned to the user program by means of a PBX instruction which, of course, by then would not work. The instruction processor, being pipelined and operating asynchronously with respect to the ALU, proceeded to decode the next three macroinstructions. Thus, the 16-bit opcode was substituted over the wrong instruction. Because instruction processor interpretation of the BKPT instruction stops the pipeline, this problem was not encountered for PBX instructions correctly paired with BKPT instructions. To correct the first side effect, the PBX instruction was redefined with the restriction that the 16-bit opcode actually fetched from the instruction stream and used as the target of a PBX must be a BKPT instruction. In this way, a policy decision eliminated the problem.

A more complicated side effect involved making the BKPT and PBX instructions work properly for breakpoints that could tolerate page faults or interrupts (eg, breakpoints during string move operations). Under some circumstances, a page fault during execution of the PBX target instruction caused execution of the BKPT instruction, after the page fault was resolved, when the PBX target instruction should have been executed. This resulted from methods used to load and restore state information on the user stack.

Implementation group members who were committed to the desirability of these debugging features formed an ad hoc team that devoted considerable energy to either properly designing the microinterpreter for this instruction pair, or eliminating the pair altogether. Because these side effects were detected late in the design cycle, an immediate solution was necessary. A reasonable and effective solution was found, one that did not further affect the instruction definition process. This solution serves as a clear example of how a first implementation can and will affect instruction set architecture.

Precise and Compatible Subroutine Handling

The 16-bit ECLIPSE computer directly supported a powerful stack mechanism that handled the stack pointer, frame pointer, and stack overflow. In the ECLIPSE MV/8000 computer, the 32-bit stack was expanded to include stack overflow and underflow detection and complete cutback of a stack frame—including passed arguments—via only one instruction, a RETURN. The 32-bit stack typically supports subroutines invoked through the CALL instruction, if one or more arguments are passed, or by means of a Jump to Subroutine (JSR) instruction, if no arguments are passed on the stack. Ultimately, the same stack frame is built in each case, and the same instruction is used for the return (Fig 10).

Precise calling conventions were desirable from two vantage points. In one case, studies revealed that the execution time for subroutine call and return instructions represents a significant portion of overall program execution time, even in highly computational floating point operations, making a fast and efficient mechanism desirable. In the other case, it was mandatory to support a 32-bit common runtime library patterned after the structure successfully developed for the ECLIPSE computer. In fact, the ECLIPSE MV/8000 computer's call, save, and return mechanism supported by the hardware accurately mimics this convention.

One noteworthy aspect of stack support is the attention given to proper handling, identification, and

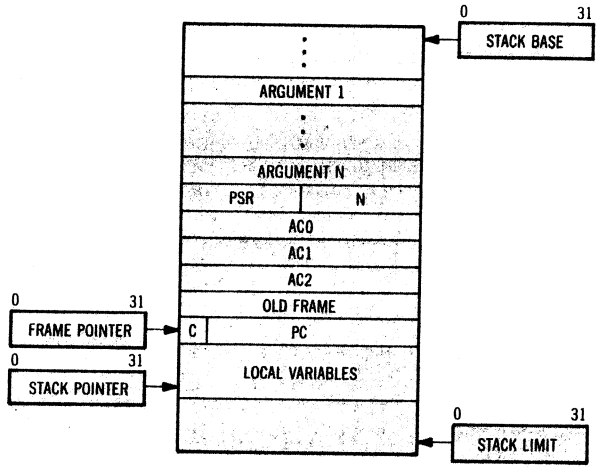


Fig 10 Stack structure. Hardware stack and frame pointers manage stack. Hardware base and limit registers detect overflow and underflow, trapping to user routine in either case. Subroutine CALL instruction passes arguments and constructs stack frame. JSR instruction builds stack frame without passing arguments. All structures support calling conventions of common runtime library

recoverability of stack faults. A test is made for stack overflow upon every operation that pushes an item onto the stack. During every operation that pops an item from the stack, a test is made for stack underflow. When a stack fault is detected, a trap occurs by vectoring through a location in page 0 of the current segment; thus, the user program—not the operating system—handles the trap.

Summary

Hardware and software implementation and architectural aspects of the ECLIPSE MV/8000 computer project give unique insights into the technical and organizational requirements for bringing a successful 32-bit minicomputer to the marketplace. Objectives must be clearly defined; what is more, they must be used to evaluate the various alternatives that inevitably arise. Within the scope of the project and the schedules set forth, all reasonable alternatives must be considered. Decisions should be made as promptly as possible, so that the implementation can proceed and the next set of alternatives can be analyzed. An indication of the success of this approach is that, in addition to achieving its primary goals, the final ECLIPSE MV/8000 computer design runs existing AOS 16-bit programs anywhere from two to three times faster than the hardware with which it was to be compatible.

References

1. ECLIPSE MV/8000, microNOVA, and SWAT are trademarks of Data General Corp, and ECLIPSE and NOVA are registered trademarks.
2. E. I. Organick, *The MULTICS System: An Examination of its Structure,* MIT Press, 1972
3. "Introduction to the Advanced Operating System (AOS)," 093-000121, Data General Corp, Westboro, Mass
4. W. W. Chu and H. Opderbeck, "Performance of Replacement Algorithms with Different Page Sizes," *Computer,* Nov 1974, pp 14-21
5. R. Morris, "Scatter Storage Techniques," *Communications of the ACM,* Jan 1968, pp 38-43
6. D. E. Knuth, *The Art of Computer Programming: Sorting and Searching,* vol 3, Addison-Wesley, 1973
7. L. A. Belady, "A Study of Replacement Algorithms for Virtual Storage Computers," *IBM Systems Journal,* vol 8, no 2, 1966, pp 78-101
8. D. J. Hatfield, "Experiments on Page Size, Program Access Patterns, and Virtual Memory Performance," *IBM Journal of Research and Development,* Jan 1972, pp 58-66
9. "IBM System/370 Principles of Operation," GA22-7000, IBM Corp, Poughkeepsie, NY
10. J. H. Saltzer and M. D. Schroeder, "The Protection of Information in Computer Systems," *Proceedings of the IEEE,* Sept 1975, pp 1278-1308
11. R. S. Fabry, "Capability-Based Addressing," *Communications of the ACM,* vol 17, no 7, July 1974, pp 403-412

About the Authors:

Steve Wallach, now the staff specialist involved with advanced system development for the ECLIPSE MV/8000 system, was principal architect for that system and manager of advanced development for ECLIPSE systems. He received a BSEE from the Polytechnic Institute of Brooklyn, an MSEE from the University of Pennsylvania, and an MBA from Boston University.

Chuck Holland, a design engineer, has headed the microcode development group for the ECLIPSE MV/8000 and participated in the development of the system's architecture and design. At present, he is working with an advanced development group. He received a BS and an MSEE from Georgia Tech.

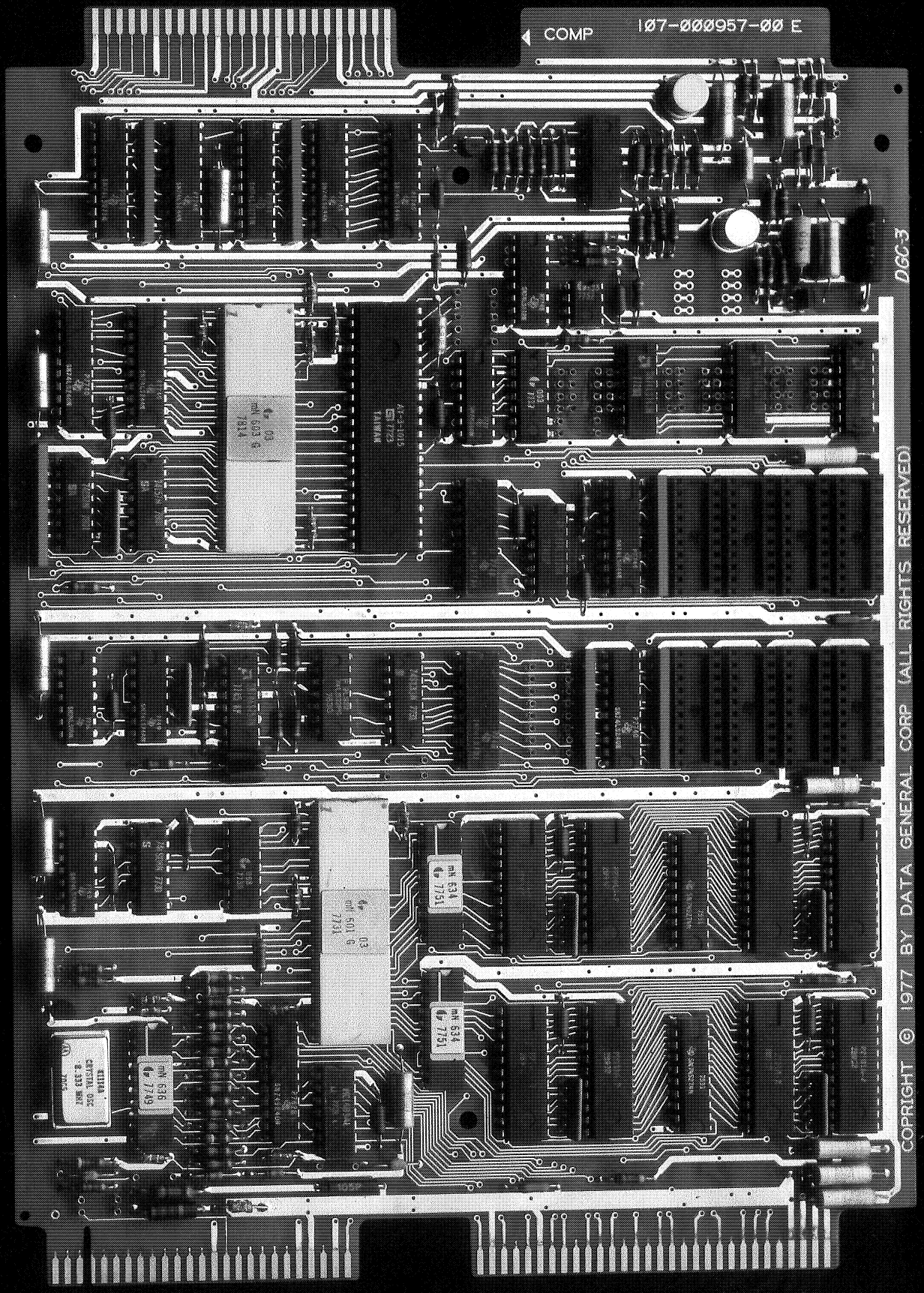
Please rate the value of this article by circling the appropriate number in the "Comments" box on the Inquiry Card.

High 701 **Average 702** **Low 703**

Data General

microNOVA Board Computer MBC/1

The Microcomputer With Full Computer Capabilities



microNOVA Board Computer MBC/1

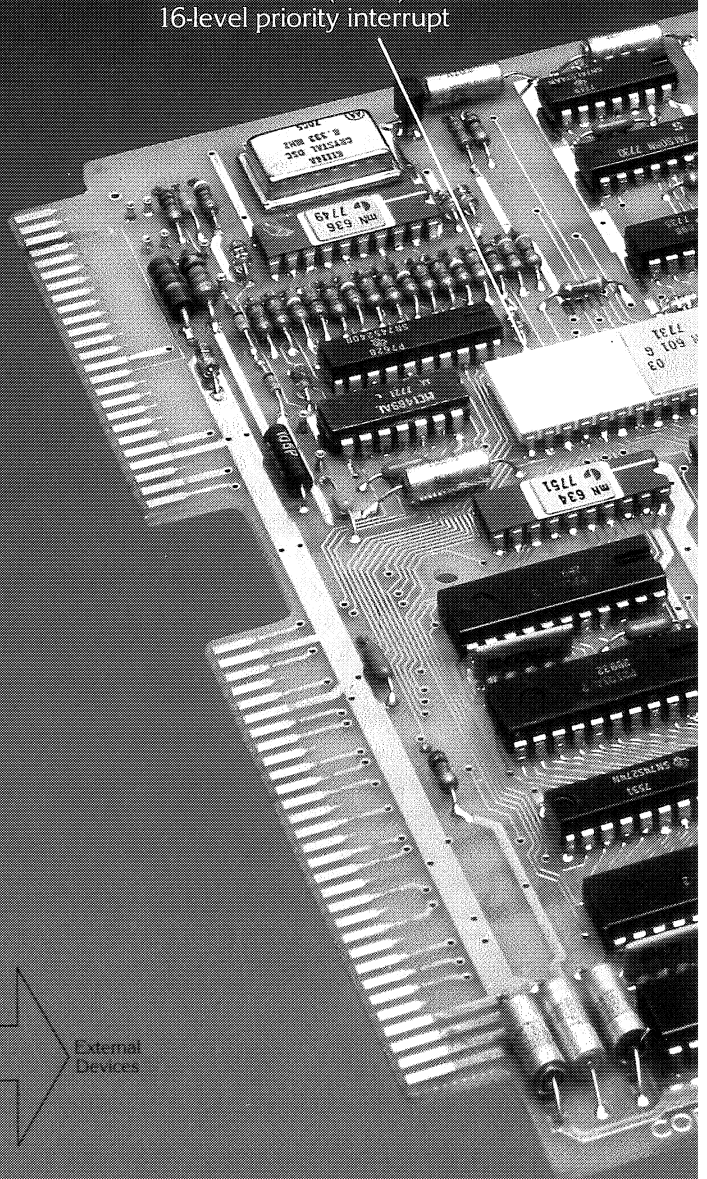
Data General's microNOVA™ Board Computer (MBC/1) combines minicomputer performance and systems capability with microcomputer board technology and economy. MBC/1 meets the growing needs of users in real-time applications such as instrumentation, industrial control, communications, intelligent controllers, and test equipment.

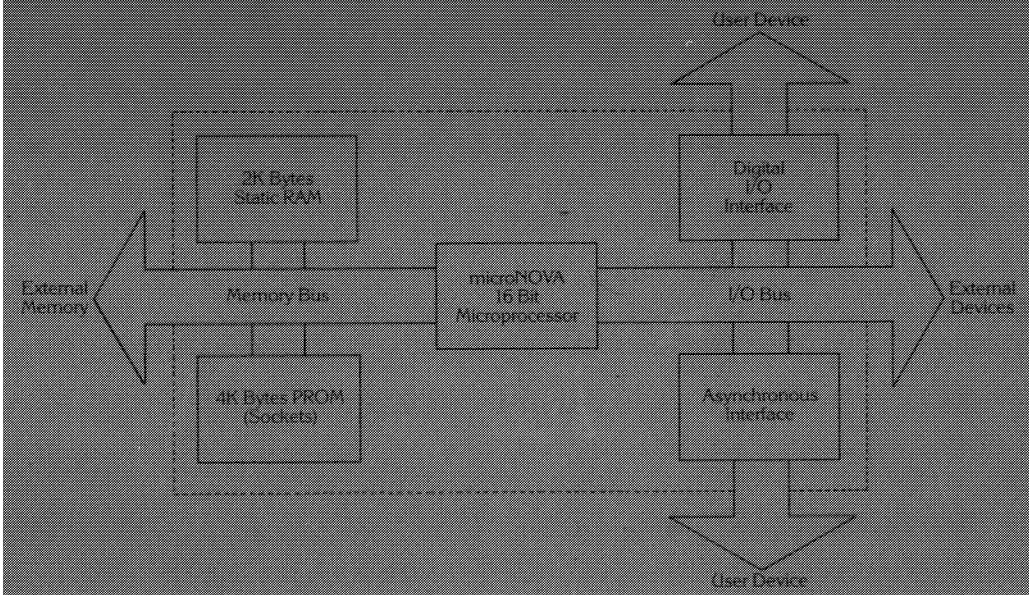
The 7.5″ x 9.5″ single-board MBC/1 contains a 16-bit microNOVA CPU with full NOVA® architecture, hardware stack and frame pointer, 16-bit multiply and divide, real-time clock, hidden memory refresh, data channel (DMA), 16-level priority interrupt, 2K-bytes of static RAM, sockets for 4K-bytes of PROM memory, an asynchronous communications interface, and a 32-line digital input/output port. Software for MBC/1 includes a multitasking monitor (MBC/M) with support for MBC/1 devices and monitor emulators allowing program development under Data General's Disc Operating System (DOS), Real-time Disc Operating System (RDOS), and Advanced Operating System (AOS). Also provided are console debug and self-test diagnostics.

The MBC/1 is mechanically and electrically compatible with the entire microNOVA product family allowing full system expansion with a complete line of Data General supplied packaging accessories and support interfaces. And microNOVA, NOVA and ECLIPSE® systems with supporting peripherals, operating systems and high-level languages provide extensive program development tools.

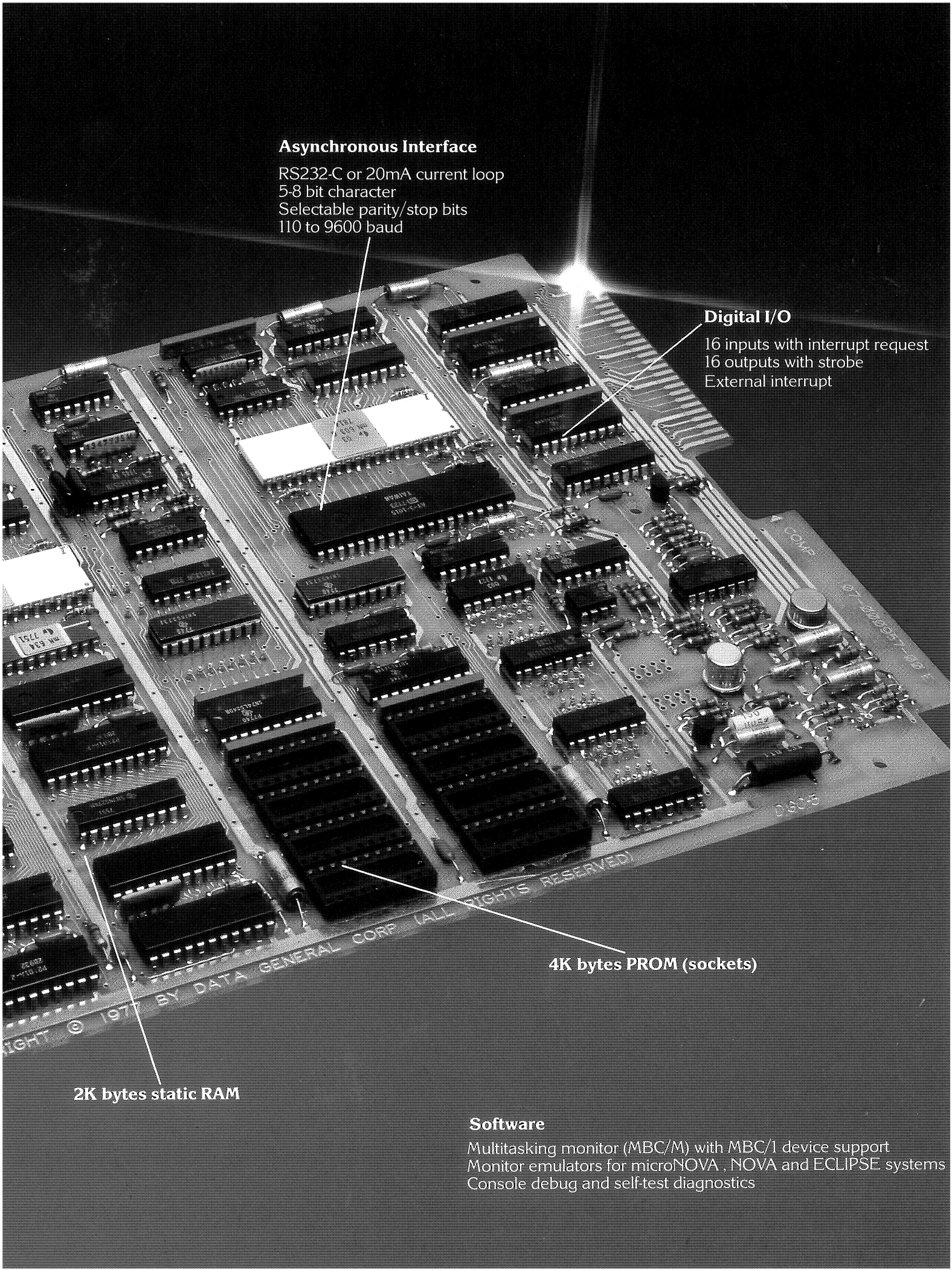
microNOVA 16-bit microprocessor

Full NOVA architecture
Hardware stack and frame pointer
16-bit multiply and divide
Real-time clock
Hidden memory refresh
Data channel (DMA) control
16-level priority interrupt





MBC/1 System Block Diagram


Asynchronous Interface
RS232-C or 20mA current loop
5-8 bit character
Selectable parity/stop bits
110 to 9600 baud
Digital I/O
16 inputs with interrupt request
16 outputs with strobe
External interrupt
4K bytes PROM (sockets)
2K bytes static RAM
Software
Multitasking monitor (MBC/M) with MBC/1 device support
Monitor emulators for microNOVA , NOVA and ECLIPSE systems
Console debug and self-test diagnostics

NORTH AMERICAN OFFICES: Westboro, Massachusetts, 01581, (617) 366-8911, headquarters. And AL: Birmingham; AZ: Phoenix, Tucson; CA: El Segundo, Palo Alto, Paramount, Sacramento, San Diego, San Francisco, Santa Ana, Santa Barbara, Van Nuys, Woodland Hills; CO: Englewood; CT: North Branford; FL: Ft. Lauderdale, Orlando, Tampa; GA: Atlanta; ID: Boise; IL: Peoria, Schaumburg; IN: Indianapolis; KY: Louisville; LA: Baton Rouge; MA: Cambridge, Springfield, Wellesley, Worcester; MD: Baltimore; MI: Southfield; MN: Minneapolis; MO: Kansas City, St. Louis; NC: Charlotte, Greensboro; NH: Nashua; NJ: Cherry Hill, Wayne; NM: Albuquerque; NV: Las Vegas; NY: Buffalo, Latham, Melville, New York City, Newfield, Rochester, Syracuse; OH: Columbus, Dayton, Euclid; OK: Oklahoma City, Tulsa; OR: Portland; PA: Blue Bell, Carnegie; RI: Albion, Rumford; SC: Columbia; TN: Knoxville, Memphis; TX: Austin, Dallas, El Paso, Houston; UT: Salt Lake City; VA: Hampton, McLean, Norfolk, Richmond, Salem; WA: Kirkland; WI: Milwaukee; CANADA: ALBERTA: Calgary, Edmonton; B.C.: Richmond; ONTARIO: Ottawa, Toronto; QUEBEC: St Laurent (Montreal)

INTERNATIONAL OFFICES: ARGENTINA: Buenos Aires; AUSTRALIA: Adelaide, Brisbane, Melbourne, Newcastle, Perth, Sydney; AUSTRIA: Vienna; BELGIUM: Brussels; BRAZIL: Sao Paulo; CHILE: Santiago; COSTA RICA: San Jose; DENMARK: Copenhagen; ECUADOR: Quito; EGYPT: Cairo; FINLAND: Helsinki; FRANCE: Paris, Le Plessis-Robinson, Lyon; GERMANY: Dusseldorf, Eschborn, Filderstadt, Hamburg, Munich, Ratingen; GREECE: Athens; HONG KONG; INDIA: Bombay; IRAN: Teheran; ISRAEL: Givatayim; ITALY: Milan, Rome; JAPAN: Tokyo; KOREA: Seoul; KUWAIT: Kuwait; LEBANON: Beirut; MALAYSIA: Kuala Lumpur; MEXICO: Mexico City; NETHERLANDS: Rijswijk; NEW ZEALAND: Auckland, Wellington; NICARAGUA: Managua; NORWAY: Oslo; PERU: Lima; PHILIPPINES: Manila; PORTUGAL: Lisbon; PUERTO RICO: Hato Rey; SAUDI ARABIA: Riyadh; SINGAPORE; SOUTH AFRICA: Capetown, Johannesburg, Pretoria; SPAIN: Barcelona, Bilbao, Madrid, San Sebastian, Valencia; SRI LANKA: Columbo; SWEDEN: Gothenburg, Malmo, Stockholm; SWITZERLAND: Lausanne, Zurich; TAIWAN: Taipei; TURKEY: Ankara; UNITED KINGDOM: Birmingham, London, Manchester, South Harrow, Glasgow; URUGUAY: Montevideo; USSR; VENEZUELA: Maracaibo.

Data General

Data General Corporation, Westboro, Massachusetts 01581, (617) 366-8911

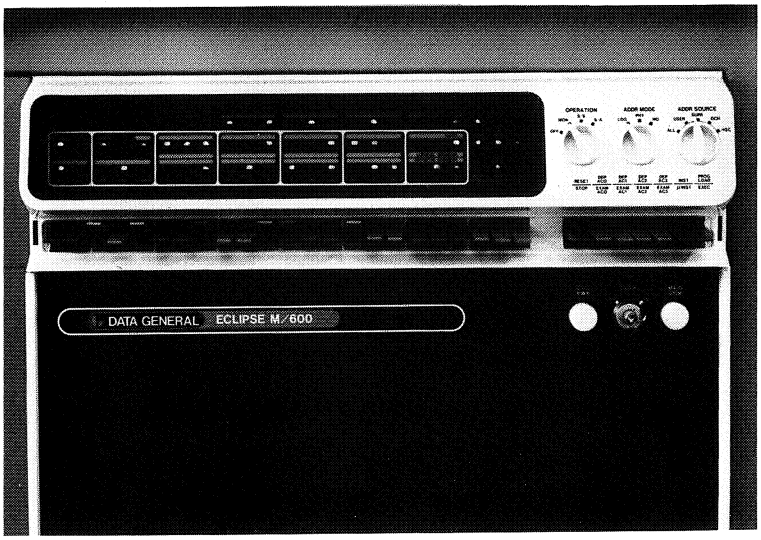
Data General

ECLIPSE M/600 COMPUTER

PRODUCT BRIEF

FEATURES

- Advanced microprogrammed architecture
- Comprehensive 16- and 32-bit instruction set
- Hardware stack with variable-length frames
- Sophisticated demand-paged memory management
- Memory expansion to one megabyte
- Semiconductor or core memory
- Three-level I/O management system (IOMS)
  -High-speed Burst Multiplexor Channel (BMC)
  -Medium-speed standard Data Channel
  -Character-oriented I/O Processor (IOP)
- Attractive, reliability-oriented packaging
- Complete software support
- Wide peripheral product line
- Versatile configurability



DESCRIPTION

The ECLIPSE M/600 computer system is the top-of-the-line ECLIPSE computer. It provides the power and versatility needed for a wide variety of applications. M/600 systems use the comprehensive ECLIPSE® instruction set for high throughput in both scientific and business data processing applications. Demand-paged memory management lets the system concurrently process multiple applications whose total storage requirements exceed physically available memory. An independent Input/Output Processor (IOP) handles all terminal input and output, eliminating the vast majority of interrupt processing on the system's Job Processor. An ultra-high-speed Burst Multiplexor Channel and the standard data channel give M/600 systems extremely high block-I/O throughput capability.

The M/600 electrical and mechanical design ensures extremely high uptime by maximizing system reliability. It also incorporates many advanced maintainability features. The computer is housed in an attractive three-bay cabinet with expansion units available for additional peripherals.

ECLIPSE M/600 system software support is based on the Advanced Operating System (AOS) with a wide variety of utilities and user programming languages. AOS supports concurrent timesharing and batch operations.

ADVANCED MICROPROGRAMMED ARCHITECTURE

ECLIPSE M/600 systems feature an advanced microprogrammed architecture in both its Job and I/O Processors. By achieving a high degree of functional parallelism in each 200-nanosecond microcycle, M/600 systems retain flexible, economical microprogrammed implementation while approaching the hardwired-logic speed. In every microcycle, internal parity hardware provides a self-check of the timing margins throughout the Job Processor.

INSTRUCTION SET

The ECLIPSE M/600 Job Processor's powerful, flexible instruction set ensures quick, efficient machine response. Instructions include both 16-bit single-word instructions for efficient memory use and 32-bit double-word instructions for extended addressing range. Absolute, relative, indexed, immediate and indirect addressing are available in both 16-bit and 32-bit instruction formats.

M/600 systems feature the powerful Standard ECLIPSE Instruction Set plus four Source-Data-Processing Instruction Set Extensions. The standard instruction set performs fixed point arithmetic; word, block, byte, bit, and bit-string manipulations; signed and unsigned interger multiply/divide and logical operations; and single- and double-word binary and hexadecimal shifts.

The two Business Source-Data-Processing Extensions include the Character Instruction Set and the Decimal/Edit Instruction Set. Character instructions move byte strings from one memory area to another, compare one byte string to another, convert one byte string from one internal representation to another, and conditionally move and test a byte string until one or more software-specified delimiters are found. Both fixed- and variable-length character data fields are supported.

The Decimal/Edit Instruction Set lets users directly process packed or unpacked decimal data and format numeric data under a hardware edit subprogram. Edit operations include leading-zero suppression, leading or trailing signs, floating decimals, punctuation control, and text insertion into a destination field.

The two Scientific Source-Data-Processing Extensions include the Floating-Point Instruction Set and the Aggregate-Operation Instruction Set. The Floating-Point Instruction Set lets users quickly process single- or double- precision floating-point data. For example, a double-precision (64-bit) Floating-Point Add takes 1.0 microsecond and a double-precision Floating-Point Divide takes 6.8 microseconds. This instruction set is also highly flexible. It has five separate hardware registers—four 64-bit accumulators and a 32-bit status register; it encompasses 56 operations, including fast single-word register-to-register instructions, and double-word memory-to-register instructions. Stack operations are supported in two ways: random indexed access into the current stack frame and block-oriented PUSH and POP instructions.

The Aggregate-Operation Instruction Set speeds scientific computation by performing complex functions such as square root, logarithms, exponentials, sines, cosines, and polynomial evaluation with single instructions. By incorporating complex functions into hardware, the Aggregate-Operation Instruction Set dramatically improves computational performance.

The M/600 instruction set is upward compatible with the standard NOVA® and ECLIPSE instruction sets.

EXTENDED HARDWARE STACK

The ECLIPSE M/600 system's extended hardware stack is a last-in, first-out stack consisting of a series of variable-length frames. The frames can be accessed randomly. A SAVE instruction saves the machine state during subroutine linkage and allocates a stack frame. A RETURN instruction reverses the procedure. PUSH and POP instructions save and restore single or multiple registers. Stack protection features automatically detect stack overflow and underflow.

DEMAND-PAGED MEMORY MANAGEMENT

ECLIPSE M/600 systems feature up to one megabyte of main system memory with advanced demand-paged memory management. The M/600 system's memory manager divides all user programs into 2K-byte pages. At any given time it stores active pages in main memory and stores other pages on high-speed devices, such as fixed head discs. Demand-paged memory techniques let systems accomodate more users because only active parts of their programs—the user's "working set"—is memory resident at any given time. All paging and page management activities are handled transparently for users by the hardware and system software.

Total main memory can include up to one megabyte of high-density semiconductor memory with 500-nanosecond read cycles and 700-nanosecond write cycles. Up to 512 K-bytes of core memory with 800-nanosecond cycles is also permitted. All semiconductor memory has Error Checking and Correction for maximum data integrity. Memory can be interleaved up to eight ways to achieve an "effective cycle time" as low as 300 nanoseconds for certain instructions, and 200-nanoseconds for burst I/O.

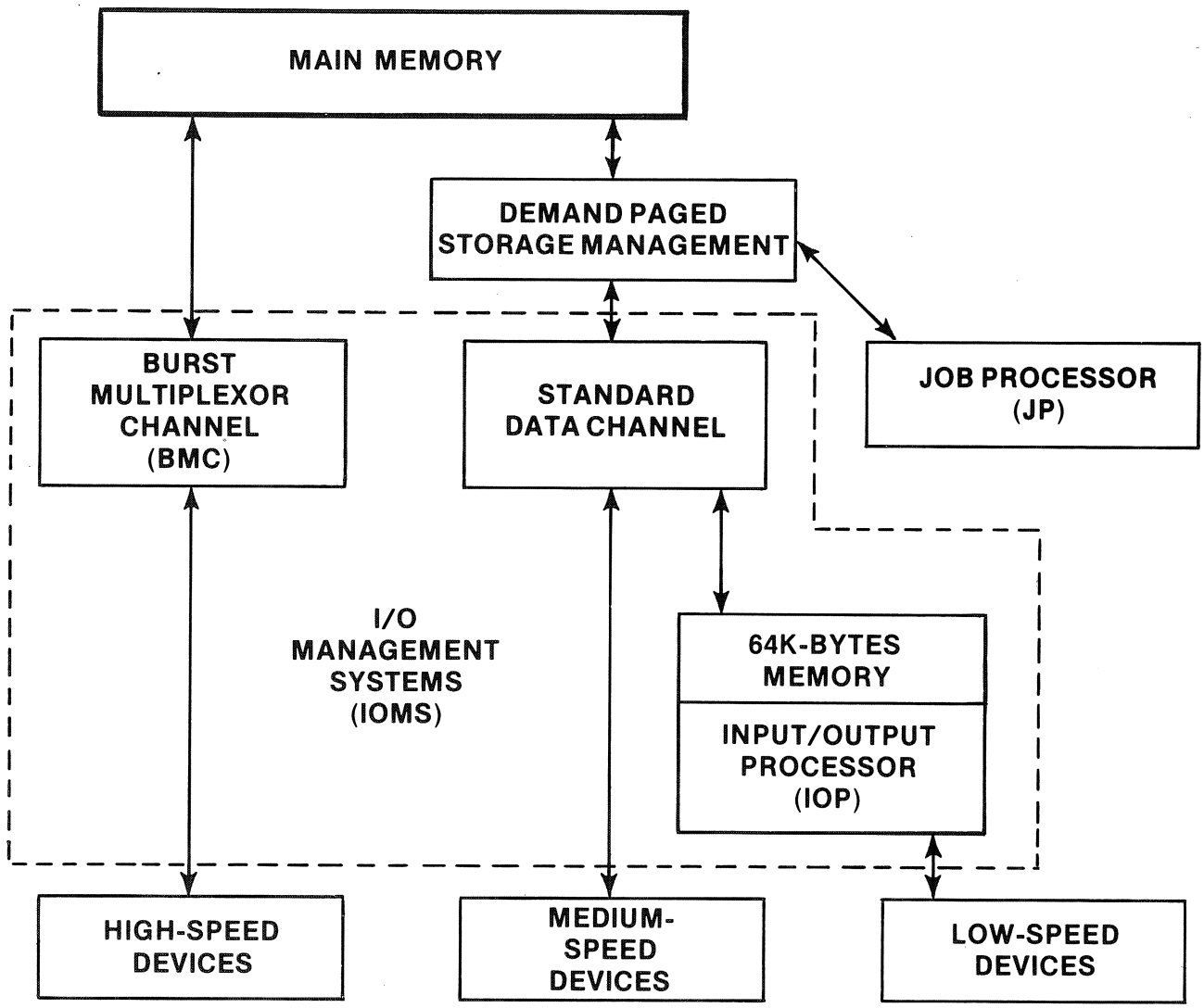
INPUT/OUTPUT MANAGEMENT SYSTEM

The ECLIPSE M/600 system's three-level Input/Output Management System (IOMS) provides optimized data-transfer characteristics for each peripheral-device class.

High-performance storage devices, such as the Fixed-Head DG/Disc Subsystems and the DG/Disc Storage Subsystems, have transfer rates in the megabyte/second range. They are serviced by the Burst Multiplexor Channel (BMC). The BMC provides an aggregate channel speed of up to 10 megabytes/second, and supports data transfer on eight high-speed controllers concurrently.

The standard Data Channel handles data transfer for medium performance devices, such as DG/Disc cartridge discs, magnetic tape subsystems, high-speed printers, and synchronous communications equipment. Data Channel throughput capacity is up to 2.5 megabytes/second.

The I/O Processor (IOP) handles large number of low-speed devices such as asynchronous terminal equipment. The IOP features a full ECLIPSE Instruction set and 64K bytes of local storage. It can also "map" any portion of main storage into its address space, allowing transfer of logical I/O records to and from Job Processor buffer areas. The IOP effectively eliminates terminal interrupt and character-processing chores from the Job Processor, resulting in fast terminal response and high batch throughput.


MAIN MEMORY
DEMAND PAGED STORAGE MANAGEMENT
BURST MULTIPLEXOR CHANNEL (BMC)
STANDARD DATA CHANNEL
JOB PROCESSOR (JP)
I/O MANAGEMENT SYSTEMS (IOMS)
64K-BYTES MEMORY
INPUT/OUTPUT PROCESSOR (IOP)
HIGH-SPEED DEVICES
MEDIUM-SPEED DEVICES
LOW-SPEED DEVICES

SOFTWARE

The multiprogramming Advanced Operating System (AOS) and a wide variety of user programming languages are available on ECLIPSE M/600 systems. AOS lets multiple users simultaneously develop and execute programs. It has sophisticated processor, memory, mass storage, and I/O management to handle multiple concurrent timesharing and batch operations. AOS provides advanced data file handling on a variety of file structures.

Language support includes FORTRAN IV, globally optimizing FORTRAN 5, PL/I, Extended BASIC, Data General's System Programming Language (DG/L), and Macro Assembler. Utilities include HASP II and IBM 2780/3780 emulators for communications and remote job entry applications.

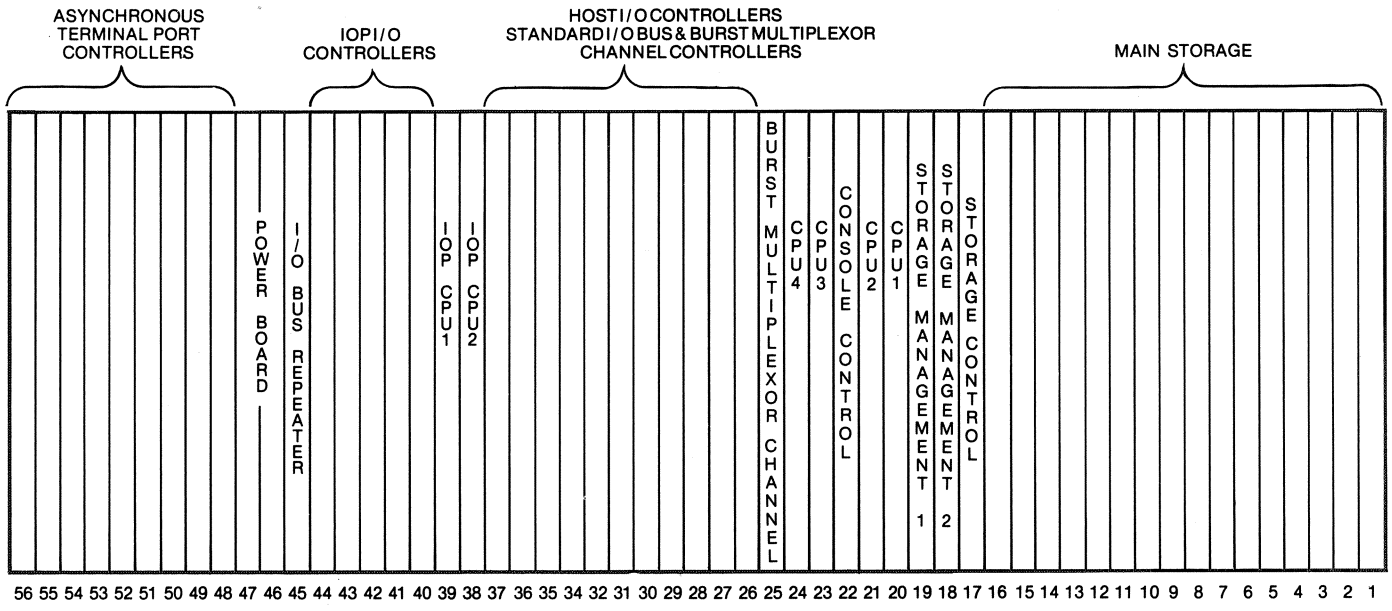
PERIPHERALS AND OPTIONS

ECLIPSE M/600 systems are available with Data General's full line of input/output devices. This includes the fixed-head disc, cartridge disc, 96- and 190-megabyte discs, diskettes, industry-compatible magnetic tape drives, DASHER™ displays and printers, line printer, and card readers. Communications equipment includes Data General's DG/CS Communications Subsystem.

CONFIGURATIONS

ECLIPSE M/600 computers use fourteen circuit boards to implement their Job Processor, demand-paged storage management, and I/O management system. A total of 56 boards can be accommodated.

The basic three-bay M/600 system is 57" (142.86cm) high, 78" (195.24cm) wide and 31" (107.78cm) deep. Additional bays can be added for extra peripherals. A service clearance of 36" (90.0cm) is required behind all cabinets.



SPECIFICATIONS

Memory Word Length: 16 bits.

Instruction Length: 16- and 32-bits.

Hardware Accumulators: 4 integer and 4 floating point.

Index Registers: 2 hardware.

Address Modes: Direct addressing of 64K-bytes; also immediate, indexed, relative and multilevel indirect addressing.

Memory Cycle Time: 800 nanoseconds (core memory), 500 nanoseconds (semiconductor memory read cycles), 700 nanoseconds (semiconductor memory write cycles).

Memory Capacity: One megabyte (semiconductor memory), 512K bytes (core memory).

Memory Increments: 64K bytes (semiconductor), 32K bytes (core).

Burst Multiplexor Channel: Maximum byte transfer rate—10.0 MHz input, 6.67 MHz output.

BMC latency: 1.0 microseconds.

Standard Data Channel: Maximum byte transfer rate—2.50 MHz input, 1.428 MHz output.

Data Channel Latency: 4.3 microseconds.

Input/Output System: 16-bit word length, 16 priority interrupt levels, 57 devices addressable.

I/O Bus Levels: Ground and +3 volts.

AC Power Requirements: 208/120 Vac, 47-63 Hz, 32A, 3-Phase.

Heat Generated: 26,400 BTU/HR.

Temperature Range: 32° to 110°F (0° to 45°C) standard operating; -30° to 158°F (-35° to 70°C) storage.

Relative Humidity Range: To 90% operating; to 95% storage.

Altitude Range: To 10,000' operating; to 50,000' storage.

Three-bay Cabinet Dimensions: 57" (142.86cm)H x 78" (195.24cm)W x 31" (107.78cm)D.

Weight: 695 lbs. (316 Kg)

Power Cable: 9' long, 5-conductor, #8 guage ending in Hubble #265-19 plug; mates to #265-16 cable or #265-21 wall receptacle.

ECLIPSE M/600 INSTRUCTION SET

FIXED-POINT ARITHMETIC INSTRUCTIONS

Add	ADD	Increment and Skip if Zero	ISZ
Add Immediate	ADI	Load Accumulator	LDA
Add Word Immediate	ADDI	Move	MOV
Compare Limits	CLM	Negate	NEG
Complement	COM	Sign Extend and Divide	DIVX
Decimal Add	DAD	Signed Divide	DIVS
Decrement and Skip if Zero	DSZ	Signed Multiply	MULS
Decimal Subtract	DSB	Skip if ACS > ACD	SGT
Exchange Accumulators	XCH	Skip if ACS ≥ ACD	SGE
Extended Decrement and Skip if Zero	EDSZ	Skip if Bit Non-Zero	SNB
Extended Increment and Skip if Zero	EISZ	Store Accumulator	STA
Extended Load Accumulator	ELDA	Subtract	SUB
Extended Store Accumulator	ESTA	Subtract Immediate	SBI
Halve	HLV	Unsigned Divide	DIV
Increment	INC	Unsigned Multiply	MUL

LOGICAL INSTRUCTIONS

Add Complement	ADC	Exclusive OR	XOR
AND	AND	Exclusive OR Immediate	XORI
And Immediate	ANDI	Inclusive OR	IOR
AND with Complemented Source	ANC	Inclusive OR Immediate	ORI

SHIFT INSTRUCTIONS

Double Hex Shift Left	DHXL	Hex Shift Left	HXL
Double Hex Shift Right	DHSR	Hex Shift Right	HXR
Double Logical Shift	DLSH	Logical Shift	LSH

BIT-MANIPULATION INSTRUCTIONS

Count Bits	COB	Set Bit to Zero	BTZ
Locate and Reset Lead Bit	LRB	Skip on Zero Bit	SZB
Locate Lead Bit	LOB	Skip on Zero Bit and Set to One	SZBO
Set Bit to One	BTO		

BYTE INSTRUCTIONS

Load Byte	LDB	Extended Load Byte	ELDB
Store Byte	STB	Extended Store Byte	ESTR

DATA MOVEMENT INSTRUCTIONS

Block Add and Move	BAM	Block Move	BLM

STACK INSTRUCTIONS

Modify Stack Pointer	MSP	Push Multiple Accumulators	PSH
Pop Block	POPB	Push Return Address	PSHR
Pop Multiple Accumulators	POP	Restore	RSTR
Pop PC and Jump	POPJ	Return	RTN
Push and Jump	PSHJ	Save	SAVE

FLOATING-POINT INSTRUCTIONS

Absolute Value	FAB	Pop Floating Point State	FPOP
Add Double (FPAC)	FAD	Push Floating Point State	FPSH
Add Double (Memory)	FAMD	Read High Word	FRH
Add Single (FPAC)	FAS	Scal	FSCAL
Add Single (Memory)	FAMS	Skip Always	FSA
Clear Errors	FCLE	Skip = 0	FSEQ
Compare Floating Point	FCMP	Skip ≥ 0	FSGE
Divide Double (FPAC)	FDD	Skip > 0	FSGT
Divide Double (Memory)	FDMD	Skip ≤ 0	FSLE
Divide Single (FPAC)	FDS	Skip < 0	FSLT
Divide Single (Memory)	FDMS	Skip ≠ 0	FSNE
Fix to AC	FFAS	Skip On No Zero Divide	FSND
Fix to Memory	FFMD	Skip On No Error	FSNER
Float from AC	FLAS	Skip On No Mantissa Overflow	FSNM
Float from Memory	FLMD	Skip On No Overflow	FSNO
Halve	FHLV	Skip On No OVF and No DVZ	FSNOD
Intergerize	FINT	Skip On No Underflow	FSNU
Load Double	FLDD	Skip On No UNF and No DVZ	FSNUD
Load Exponent	FEXP	Skip On No UNF and No OVF	FSNUO
Load Single	FLDS	Store Double	FSTD
Load Status	FLST	Store Single	FSTS
Move Floating Point	FMOV	Store Status	FSST
Multiply Double (FPAC)	FMD	Subtract Double (FPAC)	FSD
Multiply Double (Memory)	FMMD	Subtract Double (Memory)	FSMD
Multiply Single (FPAC)	FMS	Subtract Single (FPAC)	FSS
Multiply Single (Memory)	FMMS	Subtract Single (Memory)	FSMS
Negate	FNEG	Trap Disable	FTD
No Skip	FNS	Trap Enable	FTE

CHARACTER INSTRUCTIONS

Character Move	CMV	Character Translate	CTR
Character Compare	CMP	Character Move Until True	CMT

SPECIAL INSTRUCTIONS

Dispatch Absolute	DSPA	Extended Load Effective Address	ELEF
Load Writable Control Storage	LCSF	Extended Operation	XOP
Enter WCS	XOPI	Jump	JMP
Execute	XCP	Jump to Subroutine	JSR
Extended Jump	EJMP	Load Effective Address	LEF
Extended Jump to Subroute	EJSR	System Call	SCL

I/O INSTRUCTIONS

Data in A	DIA	Data Out B	DOB
Data in B	DIB	Data Out C	DOC
Data in C	DIC	I/O Skip	SKP
Data Out A	DOA	No I/O Transfer	NIO

CPU INSTRUCTIONS

Halt	HALT	I/O Reset	IORST
Interrupt Acknowledge	INTA	Mask Out	MSKO
Interrupt Disable	INTDS	Read Switches	READS
Interrupt Enable	INTEN	Vector	VCT

MEMORY MANAGEMENT INSTRUCTIONS

Load Map	LMP	Read Breakpoint Address	DIC
Pop Context Block	DPOP	Set Breakpoint Address	DOC
Select Page-Use Table and Word	DOA	Read Breakpoint Control Flags	DIB
Read Page-Use Flags	DIA	Set Breakpoint Control Flags	DOB
Clear Page-Use Flag	NIOC		

AGGREGATE OPERATION INSTRUCTIONS

Break Into Integer and Fractional Parts	FBRK	Sine	FLSINS
Square Root	FSQRS	Cosine	FLOSS
Double-Precision Square Root	FSQRD	Double Precision Sine	FSIND
Natural Logarithm	FLOGS	Double Precision Cosine	FLOSD
Double-Precision Natural Logarithm	FLOGD	Polynomial Evaluation	FPLYS
Real Exponential	FEXPS	Double-Precision Polynomial Evaluation	FPLYD
Double-Precision Real Exponential	FEXPD		

DECIMAL-EDIT INSTRUCTIONS

Load Integer	LOI	-Store In Stack	DSTK
Store Integer	STI	-Decrement And Jump If Non-Zero	DDTK
Store Integer Extended	STIX	-Insert Sign	DINS
Load Integer Extended	LDIX	-Insert Character Suppress	DINT
Intergerize	FINT	-Insert Character Once	DINC
Load Sign	LSN	-Insert Character J Times	DIMC
Edit	EDIT	-Insert Characters Immediate	DICI
-Set T to One	DSTO	-Move Alphabetics	DMVA
-Set T to Zero	DSTZ	-Move Numerics	DMVN
-Set S to One	DSSO	-Move Characters	DMVC
-Set S to Zero	DSSZ	-Move Numeric With Zero Suppression	DMVS
-Add To SI	DASI	-Move Digit With Overpunch	DMVO
-Add to DI	DADI	-Move Float	DMVF
-Add to P	DAPU	-End Float	DNDF
-Add To P Depending On T	DAPT	-End Edit	DEND
-Add To P Depending On S	DAPS		

SALES AND SERVICE

NORTH AMERICAN OFFICES: Westboro, Massachusetts, 01581, (617) 366-8911 Headquarters. And AL: Birmingham; AZ: Phoenix, Tucson; CA: El Segundo, Palo Alto, Paramount, Sacramento, San Diego, San Francisco, Santa Ana, Santa Barbara, Van Nuys, Woodland Hills; CO: Engelwood; CT: North Branford; FL: Ft. Lauderdale, Orlando, Tampa; GA Atlanta; ID: Boise; IL: Peoria, Schaumburg; IN: Indianapolis; KY: Louisville; LA: Baton Rouge; MA: Cambridge, Springfield, Wellesley, Worcester; MD: Baltimore; MI: Southfield; MN: Minneapolis; MO: Kansas City, St. Louis; NC: Charlotte, Greensboro; NH: Nashua; NJ: Cherry Hill, Wayne; NM: Albuquerque; NV: Las Vegas: NY: Buffalo, Latham, Melville, New York City, Newfield, Rochester, Syracuse; OH: Columbus, Dayton, Euclid; OK: Oklahoma City, Tulsa; OR; Portland; PA: Blue Bell, Carnegie; RI: Albion, Rumford; SC: Columbia; TN: Knoxville, Memphis; TX: Austin, Dallas, El Paso, Houston; UT: Salt Lake City; VA: Hampton, McLean, Norfolk, Richmond, Salem; WA: Kirkland; WI: Milwaukee; CANADA: Calgary, Alberta; Edmonton, Alberta; Richmond, B.C.; Ottawa, Ontario; Toronto, Ontario; St. Laurent (Montreal), Quebec.

INTERNATIONAL OFFICES: ARGENTINA: Buenos Aires; AUSTRALIA: Adelaide, Brisbane, Melbourne, Newcastle, Perth, Sydney; AUSTRIA: Vienna; BELGIUM: Brussels; BRAZIL: Sao Paulo; COLUMBIA: Bogata; COSTA RICA: San Jose; DENMARK: Copenhagen; ECUADOR: Quito; EGYPT: Cairo; FINLAND: Helsinki; FRANCE: Paris, Le Plessis Robinsons, Lyon; GERMANY: Dusseldorf, Eschborn, Filderstadt, Hamburg, Munich, Ratigen, GREECE: Athens; HONG KONG; IRAN: Teheran; ISRAEL: Tel Aviv; ITALY: Milan, Rome, JAPAN; Tokyo; KOREA: Seoul; KUWAIT: Kuwait City; LYBIA: Tripoli; MALAYSIA: Kuala Lumpur; MEXICO: Mexico City; NETHERLANDS: Rijswijk, NEW ZEALAND: Wellington; PERU: Lima; PHILLIPPINES: Makati, Metro, Manila; PORTUGAL: Lisbon; PUERTO RICO: Hato Rey; SAUDI ARABIA: Riyadh; SINGAPORE: Sri Lanka; SOUTH AFRICA: Johannesburg, Pretoria; SPAIN: Barcelona, Bilbao, San Sebastian, Valencia; SWEDEN: Gothenborg, Malmo, Stockholm; SWITZERLAND: Lausanne, Zurich; TAIWAN: Taipei; THAILAND: Bangkok; TURKEY: Ankara; UNITED KINGDOM: Birmingham; London; Manchester; Glasgow, Scotland; URUGUAY, Montevideo; USSR.

Data General

Data General Corporation, Westboro, Massachusetts 01581, (617) 366-8911

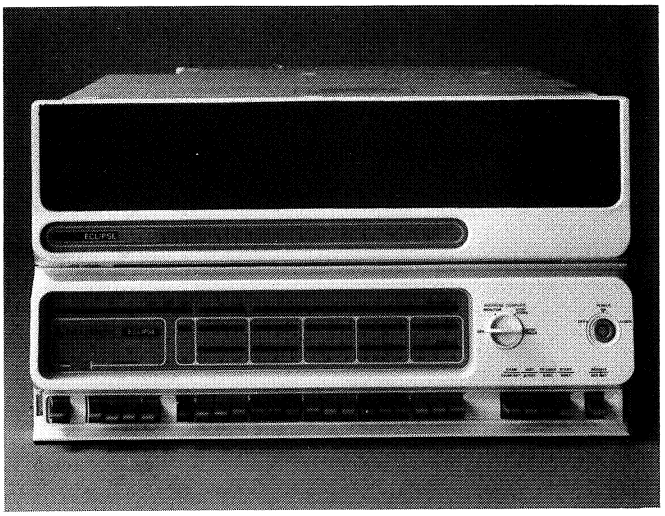
Data General

ECLIPSE AP/130 ARRAY PROCESSOR

PRODUCT BRIEF

FEATURES

- ECLIPSE® S/130 CPU with powerful array processing extensions
- Incorporates S/130 Standard Instruction Set (SIS), Floating-point Instruction Set (FIS), and Array Processing Instruction Set (APIS)
- High-speed, parallel, pipelined array processor architecture supports APIS operations on 32-bit real and 64-bit complex data
- 8K-byte, bipolar array processor memory ported to S/130 memory bus and array processor hardware
- Comprehensive set of signal processing instructions including:
  - Fast Fourier Transform (FFT)
  - Convolution
  - Correlation
  - Integration
  - Recursive Filter
  - Polynomial Evaluation
- Comprehensive set of vector/matrix processing instructions including:
  - Add/Subtract
  - Multiply
  - Compare
  - Signed Product
  - Inner Product
  - Square Magnitudes of Complex Elements
- 32-bit and 64-bit floating-point scalar arithmetic using CPU's FIS
- Memory Allocation and Protection (MAP) Unit
- Array Processor Software (APS) running under Data General's Real-time Disc Operating System (RDOS) using FORTRAN 5 or DG/L



DESCRIPTION

Data General's ECLIPSE AP/130 Array Processor provides both general-purpose scientific computing and array processing capabilities. An S/130 CPU is augmented by high-speed hardware that implements sophisticated signal and array processing instructions. The array processing hardware includes an 8K-byte 200-ns bipolar memory ported to both the array processor and the S/130 CPU. The S/130 CPU provides the Standard Instruction Set (SIS), Floating-point Instruction Set (FIS), Memory Allocation and Protection (MAP) unit, MOS/ERCC memory in 64K-byte increments, and Writable Control Store (WCS). WCS helps to implement the Array Processing Instruction Set (APIS), but a portion of WCS is available for non-APIS use.

ARCHITECTURE

AP/130 uses independent floating-point multiplier and add/subtract/compare units that operate simultaneously. This permits ultra-fast arithmetic computation. The array processor's memory includes 8K-bytes of bipolar memory with 200-nanosecond access time and 512 x 64-bit PROM that contains a sine/cosine table necessary for computing fast Fourier transforms. Array processing hardware also includes a 64-bit general-purpose working register and a 32 x 32-bit scratchpad memory. Array processor internal data transfers take place over a 64-bit data bus, which transfers data at up to 40 megabytes per second.

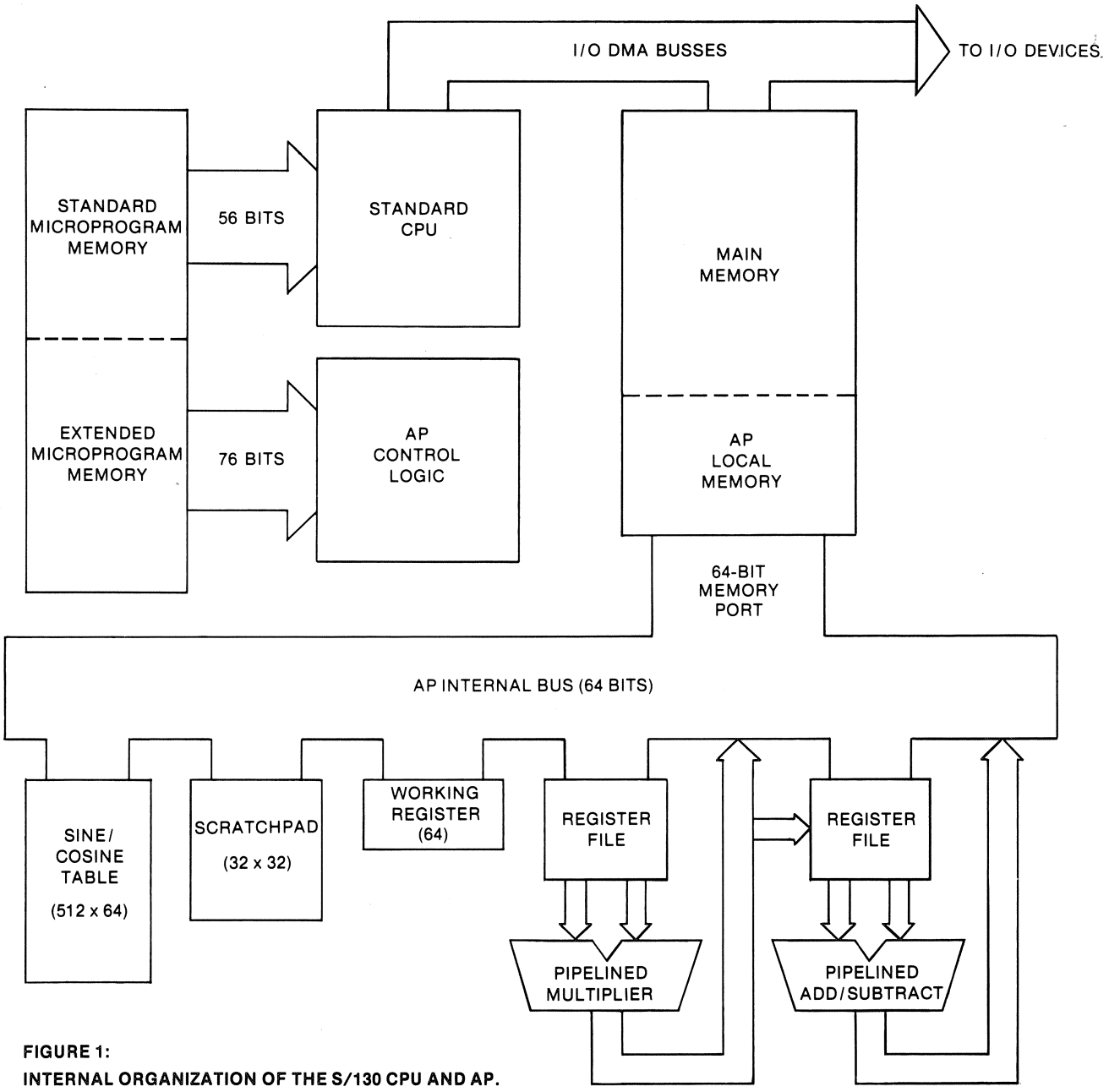


FIGURE 1:
INTERNAL ORGANIZATION OF THE S/130 CPU AND AP.

OPERATION

The AP/130 microcode is divided into two portions. The S/130 Writable Control Store (WCS) portion controls instruction flow, memory operations and condition testing. This 56-bit WCS portion is extended by 76-bit microcode extension in the array processor that controls data flow, arithmetic, and memory operations in the array processor. The 76-bit extension provides parallel, pipelined operation for maximum throughput. Because the array processor operates synchronously with the S/130, interrupt handling is not affected by the array processor. Array processor intructions are themselves interruptable with the same latency as with standard instructions.

INSTRUCTION SET/PERFORMANCE

The ECLIPSE AP/130 auguments the S/130's Standard Instruction Set (SIS) and Floating-point Instruction Set (FIS) with the comprehensive Array Processing Instruction Set (APIS). APIS instructions encompass signal and vector/ matrix processing operations and can be freely intermixed with SIS and FIS instructions. A parameter block used by each APIS instruction permits programming flexibility including the use of step registers. The S/130 Character Instruction Set (CIS) is available as an option.

The ECLIPSE AP/130 architecture permits the 46 basic APIS instructions to execute very rapidly. For example, a 1024-point complex FFT is computed in 8.75 milliseconds, a 1024-point real recursive filter (two poles, two zeros) in 1.7 milliseconds, and a 1024-point real array multiply in .65 milliseconds.

SOFTWARE

Array Processor Software (APS) runs under Data General's Real-time Disc Operating System (RDOS) using Data General's FORTRAN 5 and systems programming language, DG/L. APS lets users map array processor memory into their logical address space, execute AP instructions, and set up a control block to provide APIS instruction parameters and error handling. Users can program AP/130 in assembly language for maximum efficiency, or in FORTRAN 5 or DG/L for ease of software development.

CONFIGURATION

The ECLIPSE AP/130 contains two central processor boards, a Memory Allocation and Protection (MAP) board, three array processor boards, at least one memory board, and space for additional memory and I/O subassembly boards. The AP/130 is 10½" high, rack mountable, and has a total of 12 subassembly slots (Figure 2). Some larger configurations will require a 12-slot I/O expansion chassis, which provides additional slots for I/O subassemblies.

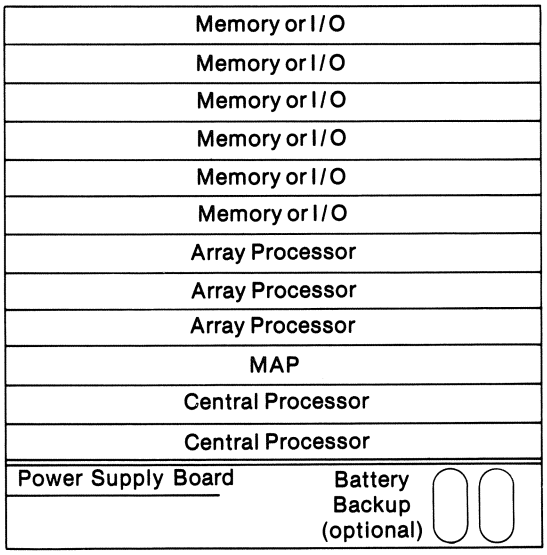


Figure 2:
Array Processor Slot Configuration

SPECIFICATIONS

Array Processor Specifications

Memory: 8K-byte 200-ns bipolar memory organized as 1024 x 64 bits; 512 x 64 bit sine/cosine table PROM; 32 x 32-bit scratchpad RAM

Register: 64-bit working register

Internal Data Bus: 64-bits wide, 40 MB/second transfer rate

S/130 CPU Specifications

Memory word length: 16 bits

Instruction length: 16-bit and 32-bit instructions

Hardware Accumulators: 4 integer plus 4 floating point

Index Registers: 2 hardware, 16 memory

Address Modes: Direct addressing of 64K bytes. Immediate, indexed, relative, and multi-level indirect addressing

Memory Type: 64K-byte, MOS/ERC, single-board modules.

Memory cycle time: 500 nanosecond read cycle, 700 nanosecond write cycle

Direct Memory Access Channel: Standard; maximum word transfer rate, 1.25 MHz, .714 MHz output

Data Channel Latency: 4.3 microseconds

Input/Output System: 16-bit word length; 16 priority interrupt levels; 59 devices addressable

I/O Bus Levels: Ground and +3 volts

Heat Generated: 6500 BTU/hr. max.

Temperature Range: 0 to 35°C standard operating; -35°C to 70°C storage

Relative Humidity Range: To 90% operating; to 95% storage

Altitude Range: To 10,000' operation; to 50,000' storage

Dimensions: 10-1/2"H x 19"W x 23-1/16"D

Weight: 145 lbs. max.

Power Cable: 6' long, wired to computer; other end Belden NEMA-type 5-15P mokled vinyl grounding plug.

ARRAY PROCESSING INSTRUCTION SET (APIS)

Scalar Instructions

ARS	Add real scalar to array
SRC	Subtract real scalar from array
MRS	Multiply real scalar by array
MCS	Multiply complex scalar by array
SPS	Signed product of real scalar and array
CMS	Compare real scalar to array

Array Instructions

ARA	Add real arrays
SRA	Subtract real arrays
MRA	Multiply real arrays
MCA	Multiply complex arrays
NRA	Negate real array
SMA	Square magnitudes of complex array
SPR	Signed product of real arrays
SER	Sum of elements of real array
PER	Product of elements of real array
PEC	Product of elements of complex array
IPR	Inner product of real arrays
IPC	Inner product of complex arrays
EPR	Evaluate polynomial in real array
EPC	Evaluate polynomial in complex array
MXR	Maximum element of real array
MNR	Minimum element of real array
CMA	Compare arrays

Signal Processing Instructions

CONR	Convolution of real arrays
CONRZ	Convolution of real arrays with zero initial conditions
CONC	Convolution of complex arrays
CONCZ	Convolution of complex arrays with zero initial conditions
CORR	Correlation of real arrays
CORC	Correlation of complex arrays
RFR	Recursive filter of real data
RFC	Recursive filter of complex data
INR	Integrate real array

Fast Fourier Transform Instructions

FFTC	Fast Fourier transform of complex array
FFTR	Fast Fourier transform of real array
BRC	Bit-reverse indices of complex array
SCB	Store complex array with bit-reversed indices

Miscellaneous Instructions

FLL	Float and load (convert integer to real)
FS	Fix and store (convert real to integer)
LSR	Load scratchpad registers
SSR	Store scratchpad registers
SRW	Store real scalar form working register
SCW	Store complex scalar form working register
LDR	Load real array (from main memory)
LDC	Load complex array (from main memory)
STR	Store real array (to main memory)
STC	Store complex array (to main memory)
CRE	Create a real array
MOD	Modify a real array

SALES AND SERVICE

NORTH AMERICAN OFFICES: Westboro, Massachusetts, 01581, (617) 366-8911, headquarters. And AL: Birmingham; AZ: Phoenix, Tucson; CA: El Segundo, Palo Alto, Paramount, Sacramento, San Diego, San Francisco, Santa Ana, Santa Barbara, Van Nuys, Woodland Hills; CO: Englewood; CT: North Branford; FL: Ft. Lauderdale, Orlando, Tampa; GA: Atlanta; ID: Boise; IL: Peoria, Schaumburg; IN: Indianapolis; KY: Louisville; LA: Baton Rouge; MA: Cambridge, Springfield, Wellesley, Worcester; MD: Baltimore; MI: Southfield; MN: Minneapolis; MO: Kansas City, St. Louis; NC: Charlotte, Greensboro; NH: Nashua; NJ: Cherry Hill, Wayne; NM: Albuquerque; NV: Las Vegas; NY: Buffalo, Latham, Melville, New York City, Newfield, Rochester, Syracuse; OH: Columbus, Dayton, Euclid; OK: Oklahoma City, Tulsa; OR: Portland; PA: Blue Bell, Carnegie; RI: Albion, Rumford; SC: Columbia; TN: Knoxville, Memphis; TX: Austin, Dallas, El Paso, Houston; UT: Salt Lake City; VA: Hampton, McLean, Norfolk, Richmond, Salem; WA: Kirkland; WI: Milwaukee; CANADA: ALBERTA: Calgary, Edmonton; B.C.: Richmond; ONTARIO: Ottawa, Toronto; QUEBEC: St. Laurent (Montreal).

INTERNATIONAL OFFICES: ARGENTINA: Buenos Aires; AUSTRALIA: Adelaide, Brisbane, Melbourne, Newcastle, Perth, Sydney; AUSTRIA: Vienna; BELGIUM: Brussels; BRAZIL: Sao Paulo; CHILE: Santiago; COSTA RICA: San Jose; DENMARK: Copenhagen; ECUADOR: Quito; EGYPT: Cairo; FINLAND: Helsinki; FRANCE: Paris, Le Plessis-Robinson, Lyon; GERMANY: Dusseldorf, Eschborn, Filderstadt, Hamburg, Munich, Ratingen; GREECE: Athens; HONG KONG; INDIA: Bombay; IRAN: Teheran; ISRAEL: Givatayim; ITALY: Milan, Rome; JAPAN: Tokyo; KOREA: Seoul; KUWAIT: Kuwait; LEBANON: Beirut; MALAYSIA: Kuala Lumpur; MEXICO: Mexico City; NETHERLANDS: Rijswijk; NEW ZEALAND: Auckland, Wellington; NICARAGUA: Managua; NORWAY: Oslo; PERU: Lima; PHILIPPINES: Manila; PORTUGAL: Lisbon; PUERTO RICO: Hato Rey; SAUDI ARABIA: Riyadh; SINGAPORE; SOUTH AFRICA: Capetown, Johannesburg, Pretoria; SPAIN: Barcelona, Bilbao, Madrid, San Sebastian, Valencia; SRI LANKA: Columbo; SWEDEN: Gothenburg, Malmo, Stockholm; SWITZERLAND: Lausanne, Zurich; TAIWAN: Taipei; TURKEY: Ankara; UNITED KINGDOM: Birmingham, London, Manchester, South Harrow, Glasgow; URUGUAY: Montevideo; USSR; VENEZUELA: Maracaibo.

Data General

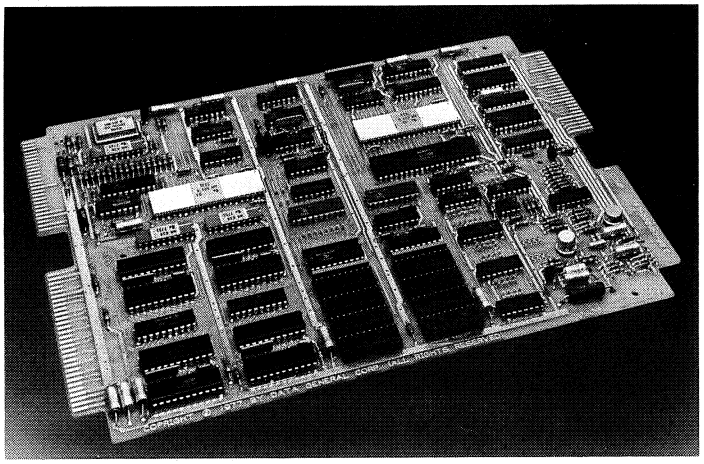
Data General Corporation, Westboro, Massachusetts 01581, (617) 366-8911

DataGeneral

microNOVA BOARD COMPUTER MBC/1

FEATURES

- Microprocessor, memory and interfaces on a single 7.5" x 9.5" board
- A full 16-bit microNOVA™ microprocessor
  - Full NOVA® architecture
  - Hardware stack and frame pointer
  - 16-Bit hardware multiply and divide
  - Real-time clock
  - Data Channel (DMA) control
  - 16-Level priority interrupt
- 2K-Bytes static RAM
- Sockets for up to 4K-Bytes PROM
- Asynchronous communications interface
- 32-line digital I/O interface
- Multitasking software support package (MBC/M)
  - Emulators for program development under all Data General's operating systems
  - Monitor for program execution on MBC/1
- Board-resident console debug software and self-test diagnostics



DESCRIPTION

Data General's microNOVA Board Computer (MBC/1) combines minicomputer performance and systems capability with microcomputer board technology and economy. The 7.5" x 9.5" single-board MBC/1 contains a 16-bit microNOVA CPU with full NOVA architecture, hardware stack and frame pointers, 16-bit multiply and divide, real-time clock, hidden memory refresh, data channel (DMA) control, 16-level priority interrupt, 2K-bytes of static RAM, sockets for up to 4K-bytes of PROM memory, an asynchronous communications interface, and a 32-line digital input/output interface (see figure 1).

A multitasking software support package (MBC/M) provides an emulator for other Data General systems that lets MBC/1 users develop programs under Data General's Advanced Operating System (AOS), Disc Operating System (DOS), and Real-time Disc Operating System (RDOS) and a monitor for program execution on MBC/1. MBC/1 also provides an optional on-board ROM console debug and self-test diagnostics.

MBC/1 is electrically and mechanically compatible with the entire microNOVA product family. This compatibility permits full system expansion with a complete line of Data General packaging accessories and support interfaces.

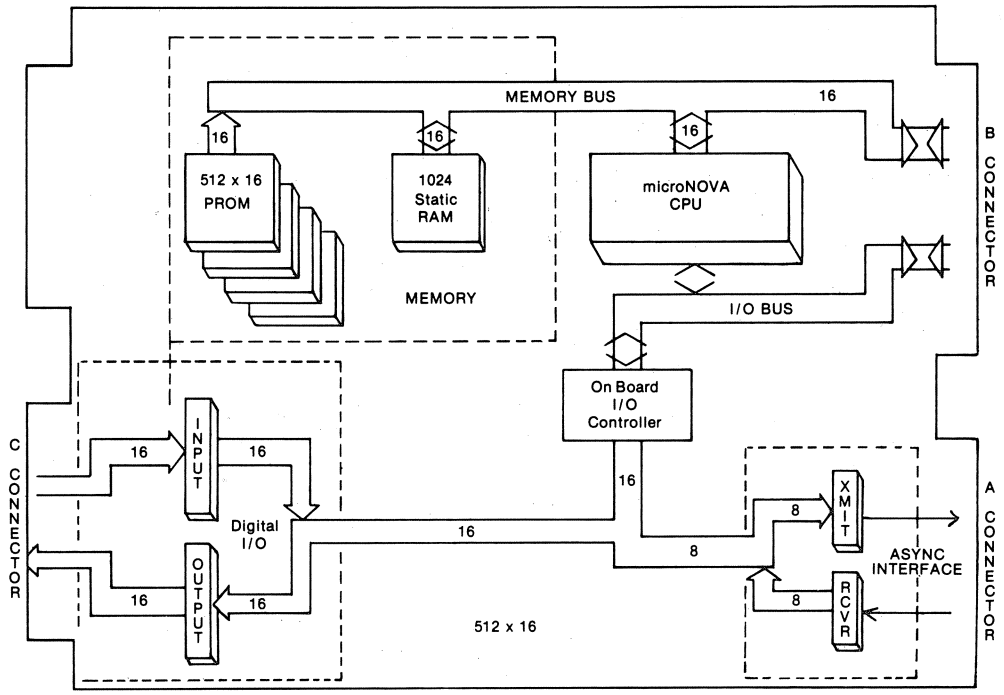


FIGURE 1

NOVA and ECLIPSE are registered trademarks of Data General Corporation.
DASHER and microNOVA are trademarks of Data General Corporation.

mN601 CENTRAL PROCESSING UNIT

The microNOVA mN601 CPU features the NOVA 16-bit multifunction instruction set, including hardware multiply/divide; multiple addressing modes including absolute, relative, indexed, deferred, and auto increment/decrement; multiple accumulators, including two that can be used as index registers, hardware stack and frame pointers with stack overflow protection; programmed priority interrupt to 16 levels; and separate memory and input/output busses. A Real-Time Clock and Random Access Memory refresh control are an integral part of the central processing unit. It also incorporates all CPU and memroy control for per-device Data Channel (DMA) transfer.

Programmed priority interrupt enables real-time response to random events. Hardware stacks facilitate re-entrant and recursive subroutine programming. Multiple accumulators reduce program size and multiple addressing modes ensure efficient memory use. The powerful microNOVA intruction set increases programmer productivity.

MEMORY

The MBC/1 provides two types of memory: Random Access Memory (RAM) and Programmable Read-Only Memory (PROM). The board includes 2K bytes of static RAM and sockets for up to 4K bytes of PROM. Industry-compatible 512 x 8 PROM let users configure programs in 512-word (1K-byte) increments. The memory bus extends off the board; this lets the user add additional memory up to 64K bytes. The MBC/1 memory bus is compatible with the standard microNOVA memory boards, allowing use of standard microNOVA RAM and PROM boards.

DIGITAL I/O INTERFACE

The MBC/1 digital I/O interface consists of 16 input lines, an external interrupt line, 16 output lines, a data strobe, and a system reset line. The input and output are TTL-compatible and use positive logic.

ASYNCHRONOUS COMMUNICATION INTERFACE

The MBC/1 asynchronous interface provides full-duplex communication between the MBC/1 and an asynchronous terminal. The devices are connected to the MBC/1 by either a 20mA current loop or EIA RS-232/C interface. Line speed (110-9600), data bits (5-8), parity (odd, even, none), and stop bits (1 and 2) are jumper selectable. In addition, three modem control signals are supported: clear-to-send, carrier detect, and data set ready.

SOFTWARE

Software support for the MBC/1 ensures easy program development using either microNOVA development systems, NOVA 3 systems, or ECLIPSE® systems. A program development library, used with Data General's Advanced Operating System (AOS), Disc Operating System (DOS) or Real-time Disc Operating System (RDOS), provides program debugging for the MBC/1.

A multitasking software support package (MBC/M) provides an emulator for program development under AOS, RDOS, or DOS and a monitor for program execution on MBC/1.

Users write and debug their application programs on the development computer using the MBC/M emulator. A library of routines provide support for all the on-board features of the MBC/1 (digital I/O and asynchronous interface). This minimizes the programming effort required to implement applications using MBC/1. Once application programs are developed and linked with the MBC/M multitasking monitor, a utility program is used to generate tapes for commercial PROM burners. Users simply plug their application PROMs into the MBC/1 to complete system integration.

The optional console debug/diagnostic program provides the user with two 512 x 8 PROMs that plug directly into PROM sockets on the MBC/1. The program provides console debug, program control, modification of program and registers, and self-test diagnostics for the MBC/1.

microNOVA FAMILY SUPPORT

The MBC/1 is electrically and mechanically compatible with the entire microNOVA line. It can be used with standard microNOVA chassis, peripherals, interfaces, and software.

Memory Boards. 8K- and 16K-byte RAM boards support the microNOVA 64K-byte memory capacity. These boards feature NMOS technology, fast access time, and refresh control from the microNOVA CPU. Additional 1K-, 2K-, 4K-, and 8K-byte PROM modules are used in applications where programs or data must be fixed permanently within the system A PROM programmer board permits PROM programming at the board level under program control.

Interface Boards. An asynchronous interface board connects DASHER™ printers and DASHER video displays to the MBC/1 I/O bus. It also provides full modem control, including automatic answer capability. A general-purpose interface board provides a generalized programmed I/O, program interrupt, and data channel interface plus predrilled areas for user-designed and built circuitry. For control applications on the microNOVA there are three boards: an A/D Interface Board, a D/A Interface Board and a Digital I/O Board. Communications is supported on the microNOVA by a four-line Asynchronous Multiplexor, a single-line Synchronous Controller, a Communications Adapter and a Cyclic Redundancy Check option.

Software. The microNOVA family is supported by powerful Data General software including the Disc Operating System (DOS), with program development utilities, and the Real-Time Operating System (RTOS). Runtime I/O support is provided by the Sensor Access Manager (SAM) for control applications and by the Communications Access Manager (CAM) for communications applications.

Packaged/Development Systems. microNOVA packaged/development systems complement the microNOVA family. They feature an MOS microNOVA minicomputer with 32K-bytes of memory, automatic program load, power fail/auto restart, terminal and dual-diskette subsystem or cartridge disc subsystem. The terminal can be any of the available DASHER video displays or printers. The diskette subsystem provides 315K-bytes (single drive) or 630K-bytes (dual drive) of online storage, and includes an integral Data Channel (DMA) controller.

The disc subsystem provides 10M bytes (5 fixed, 5 removable) of online cartridge storage and includes an integral Data Channel (DMA) controller.

The microNOVA packaged/development system operates under Data General's Disc Operating System (DOS). DOS is supported by a Command Line Interpreter, Text Editor, Macro Assembler, Library File Editor, Relocatable Loader, FORTRAN IV Compiler, single or multi-user BASIC and Business BASIC.

APPLICATIONS

Low cost, packaging flexibility and 16-bit computing power make MBC/1 ideal for many dedicated real-time applications in instrumentation, communications, data acquisition, industrial automation, and data systems markets.

Typical uses of MBC/1 in real-time dedicated control and real-time distributed control are shown below.

Real-Time Dedicated Control

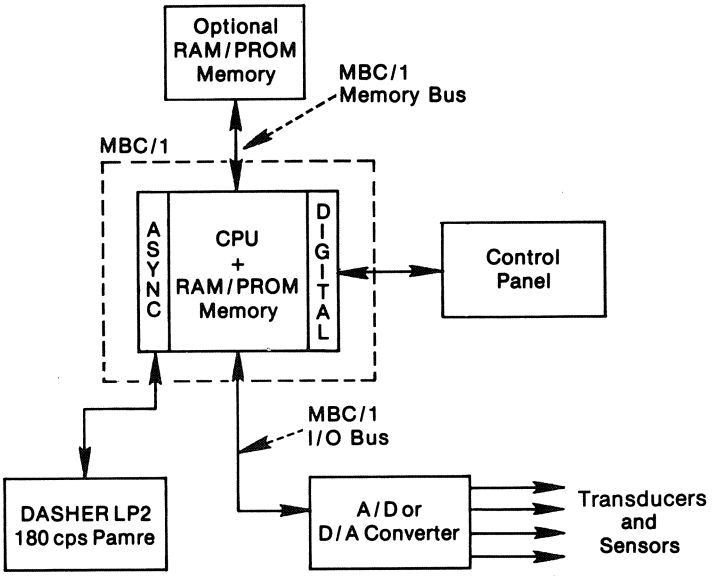


FIGURE 2

Real-Time Distributed Control

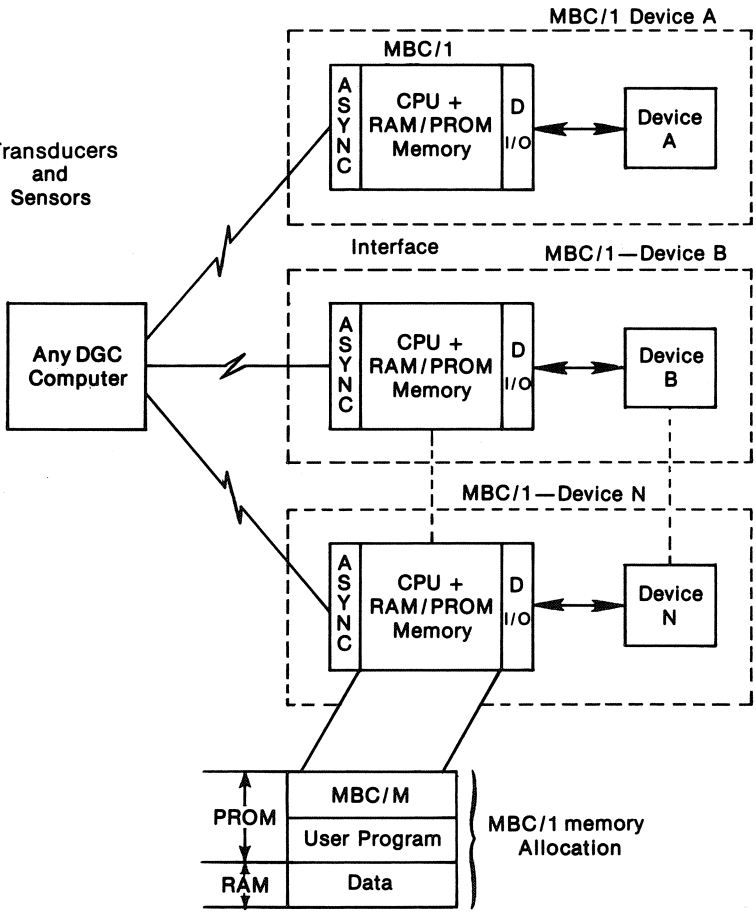


FIGURE 3

MBC/1 microNOVA BOARD COMPUTER SPECIFICATIONS

microNOVA 16-bit microprocessor: full NOVA architecture, hardware stack and frame pointer, 16-bit multiply and divide, real-time clock, hidden memory refresh, data channel (DMA) control, and 16-level priority interrupt.

Memory: 2K bytes of stactic RAM, 4K bytes PROM (sockets), expandable to 65K bytes of RAM/PROM off-board.

Asynchronous Interface: RS-232/C with modem control or 20mA current loop 5-8 bit character, selectable parity (odd, even, none)/stop bits (1,2), line speeds from 110 to 9600 BAUD.

Digital I/O Interface: 16 inputs with interrupt request line 16 outputs with strobe.

Operating Temperature: 0° to 50°C.

Power: +5V, 2.5 amps; -5V, .05 amps; +12V, .2 amps or +15V, .3 amps.

Physical Characteristics: Length 7.5 inches (19.05 cm), Width (with connectors) 10.4 inches (26.42 cm), Height .4 inches (1.02 cm), Weight 12 ounces (340.20 grams).

SALES AND SERVICE

NORTH AMERICAN OFFICES: Westboro, Massachusetts, 01581, (617) 366-8911, headquarters. And AL: Birmingham; AZ: Phoenix, Tucson; CA: El Segundo, Palo Alto, Paramount, Sacramento, San Diego, San Francisco, Santa Ana, Santa Barbara, Van Nuys, Woodland Hills; CO: Englewood; CT: North Branford; FL: Ft. Lauderdale, Orlando, Tampa; GA: Atlanta; ID: Boise; IL: Peoria, Schaumburg; IN: Indianapolis; KY: Louisville; LA: Baton Rouge; MA: Cambridge, Springfield, Wellesley, Worcester; MD: Baltimore; MI: Southfield; MN: Minneapolis; MO: Kansas City, St. Louis; NC: Charlotte, Greensboro; NH: Nashua; NJ: Cherry Hill, Wayne; NM: Albuquerque; NV: Las Vegas; NY: Buffalo, Latham, Melville, New York City, Newfield, Rochester, Syracuse; OH: Columbus, Dayton, Euclid; OK: Oklahoma City, Tulsa; OR: Portland; PA: Blue Bell, Carnegie; RI: Albion, Rumford; SC: Columbia; TN: Knoxville, Memphis; TX: Austin, Dallas, El Paso, Houston; UT: Salt Lake City; VA: Hampton, McLean, Norfolk, Richmond, Salem; WA: Kirkland; WI: Milwaukee; CANADA: ALBERTA: Calgary, Edmonton; B.C.: Richmond; ONTARIO: Ottawa, Toronto; QUEBEC: St. Laurent (Montreal).

INTERNATIONAL OFFICES: ARGENTINA: Buenos Aires; AUSTRALIA: Adelaide, Brisbane, Melbourne, Newcastle, Perth, Sydney; AUSTRIA: Vienna; BELGIUM: Brussels; BRAZIL: Sao Paulo; CHILE: Santiago; COSTA RICA: San Jose; DENMARK: Copenhagen; ECUADOR: Quito; EGYPT: Cairo; FINLAND: Helsinki; FRANCE: Paris, Le Plessis-Robinson, Lyon; GERMANY: Dusseldorf, Eschborn, Filderstadt, Hamburg, Munich, Ratingen; GREECE: Athens; HONG KONG; INDIA: Bombay; IRAN: Teheran; ISRAEL: Givatayim; ITALY: Milan, Rome; JAPAN: Tokyo; KOREA: Seoul; KUWAIT: Kuwait; LEBANON: Beirut; MALAYSIA: Kuala Lumpur; MEXICO: Mexico City; NETHERLANDS: Rijswijk; NEW ZEALAND: Auckland, Wellington; NICARAGUA: Managua; NORWAY: Oslo; PERU: Lima; PHILIPPINES: Manila; PORTUGAL: Lisbon; PUERTO RICO: Hato Rey; SAUDI ARABIA: Riyadh; SINGAPORE; SOUTH AFRICA: Capetown, Johannesburg, Pretoria; SPAIN: Barcelona, Bilbao, Madrid, San Sebastian, Valencia; SRI LANKA: Columbo; SWEDEN: Gothenburg, Malmo, Stockholm; SWITZERLAND: Lausanne, Zurich; TAIWAN: Taipei; TURKEY: Ankara; UNITED KINGDOM: Birmingham, London, Manchester, South Harrow, Glasgow; URUGUAY: Montevideo; USSR; VENEZUELA: Maracaibo.

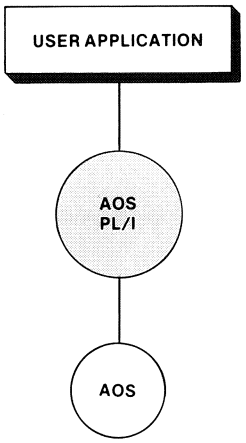
DataGeneral

Data General Corporation, Westboro, Massachusetts 01581, (617) 366-8911

PRODUCT BRIEF

FEATURES

- Implements the PL/I language based on ANSI standards
- Operates on Advanced Operating System (AOS) based ECLIPSE® systems
- Allocates storage and adjusts array sizes dynamically
- Interfaces to AOS system calls and Assembly Language routines
- Allows three levels of optimization during compilation
- Provides clear English-language diagnostic messages
- Optionally uses the ECLIPSE commercial and character instruction sets



DESCRIPTION

Data General's AOS PL/I brings large system PL/I features to the timesharing environment of AOS-based ECLIPSE systems. PL/I incorporates features of business (COBOL), scientific (FORTRAN) and systems (ALGOL) programming languages: string manipulation and data file handling capabilities for business programming, mathematical functions for the computational user, and dynamic storage allocation with bit manipulation and linked list addressing for the systems programmer implementing specialized system software.

PL/I allows structured programming, which speeds debugging and testing and reduces software development costs. Unlike most PL/I implementations, multiple users can develop, debug and execute AOS PL/I programs in an online environment using AOS timesharing facilities. Programs can write to the terminal and read data keyed in at the terminal.

ANSI PL/I SUBSET WITH EXTENSIONS

AOS PL/I is a large subset of American National Standard Programming Language PL/I (X3.53-1976). Programs written in AOS PL/I should readily compile under any standard PL/I compiler. And most PL/I programs written for other compilers can be easily modified to compile under Data General's AOS PL/I.

Extensions include convenience features such as the % REPLACE statement to define symbolic parameters, the SIZE built-in function to determine a data item's storage requirement, and the DO CASE statement for inline coding of alternate processing paths.

AOS SUPPORT

Data General' multiprogramming Advanced Operating System (AOS) controls multiple user and system tasks simultaneously. The AOS PL/I compiler operates as a shared process in an AOS programming environment, eliminating the need for separate copies of the compiler for each user. Also, all AOS PL/I compiled and library programs are re-entrant, allowing multiple users to share a single copy of program code.

DYNAMIC STORAGE ALLOCATION

AOS PL/I lets programmers dynamically determine when to allocate storage for variables, structures and arrays. Adjustable array extents and string lengths can be used for both AUTOMATIC and BASED data. AOS PL/I allocates AUTOMATIC storage when a block is entered and frees it when a block is terminated. It also allocates a new generation of AUTOMATIC storage for each recursion through a procedure. ALLOCATE and FREE statements provide complete programmer control over allocation of BASED storage.

AOS SYSTEM CALLS AND ASSEMBLY LANGUAGE INTERFACES

The SYS function lets AOS PL/I programmers directly issue AOS system calls from PL/I programs. This allows PL/I programs to delete files, access AOS system values, and perform such functions as accessing command line values and switches and interprocess communication. Data General provides a file defining the symbolic parameters used in AOS system calls with the AOS PL/I compiler.

AOS PL/I programs can easily interface to special-purpose Assembler routines through the CALL statement or function references. Also, Assembly language programs can call AOS PL/I routines.

THREE-LEVEL OPTIMIZATION

AOS PL/I offers three optimization levels at compilations. Specifying no optimization is useful for syntax checking and program debugging. Intermediate levels of optimization eliminate redundant computations and optimize branches and logical expressions. Highest-level optimization moves invariant computations outside of loops and generates the most efficient run-time code. Optionally, out-of-bounds subscripts and string ranges can be checked at runtime.

CLEAR ENGLISH-LANGUAGE DIAGNOSTICS

AOS PL/I features clear English-language diagnostics at both compile and execution time. The compiler thoroughly checks syntax during compilation and makes "reasonable" assumptions when errors are found , yielding the "cleanest" compilation possible. AOS PL/I compiler provides over 200 compiler error messages, most of which indicate both the line containing the error and an English-language description of the error.

Programs can intercept run-time errors using PL/I ON conditions. In the absence of program-specified error handling, the run-time system provides an English-language description of the error.

HARDWARE REQUIREMENTS

AOS PL/I runs on ECLIPSE computers on which AOS is licensed to run with at least 192K bytes of memory and the Floating Point Unit or Floating Point Instruction Set.

AOS PL/I programs can use the ECLIPSE Commercial and Character Instruction Sets, if available. Otherwise, the functions of these instruction sets are emulated through the software.

STATEMENTS AND OPTIONS

ALLOCATE
assignment
  element
  array
  structure
BEGIN
CALL
CLOSE
DECLARE
DELETE
DO
  simple
  increment
  list
  WHILE
  CASE[1]
END
ENTRY
FORMAT
FREE
GET
  LIST
  EDIT
  DO
  SKIP
GO TO
IF
null
ON
OPEN
PROCEDURE
  RECURSIVE
  RETURNS
PUT
  expressions
  PAGE
  LIST
  EDIT
  DO
  SKIP
READ
RETURN
  file
  entry
  label
  VARYING strings
REVERT
REWRITE
SIGNAL
STOP
WRITE

[1] Extension to ANSI Standard

BUILT-IN FUNCTIONS AND PSEUDO-VARIABLES

ABS
ACOS
ADDR
ASCII[1]
ASIN
ATAN
ATAND
BINARY
BIT
BOOL
CEIL
CHARACTER
COLLATE
COS
COSD
COSH
DATE
DECIMAL
DIMENSION
DIVIDE
EXP
FIXED
FLOAT
FLOOR
HBOUND
INDEX
LBOUND
LENGTH
LINENO
LOG
LOG2
LOG10
MAX
MIN
MOD
NULL
ONCODE
ONFILE
PAGENO
PAGENO PV
RANK[1]
ROUND
SIGN
SIN
SIND
SINH
SIZE[1]
SQRT
STRING
STRING PV
SUBSTR
SUBSTR PV
TAN
TAND
TANH
TIME
TRANSLATE
TRUNC
UNSPEC
UNSPEC PV
VALID
VERIFY

[1] Extension to ANSI standard

ATTRIBUTES

ALIGNED
AUTOMATIC
BASED
BINARY
BIT
BUILTIN
CHARACTER
DECIMAL
DEFINED
DIMENSION
  expression
  REFER
DIRECT
ENTRY
EXTERNAL
FILE
FIXED
FLOAT
INITIAL
INPUT
INTERNAL
KEYED
LABEL
length
  expression
  REFER
OUTPUT
parameter
  simple arrays (*)
  string length (*)
  structure arrays (*)
PICTURE
POINTER
precision
PRINT
RECORD
RETURNS
SEQUENTIAL
STATIC
STREAM
structure
UPDATE
VARIABLE
VARYING

SALES AND SERVICE

NORTH AMERICAN OFFICES: Westboro, Massachusetts, 01581, (617) 366-8911 Headquarters. And AL: Birmingham; AZ: Phoenix, Tucson; CA: El Segundo, Palo Alto, Paramount, Sacramento, San Diego, San Francisco, Santa Ana, Santa Barbara, Van Nuys, Woodland Hills; CO: Engelwood; CT: North Branford; FL: Ft. Lauderdale, Orlando, Tampa; GA Atlanta; ID: Boise; IL: Peoria, Schaumburg; IN: Indianapolis; KY: Louisville; LA: Baton Rouge; MA: Cambridge, Springfield, Wellesley, Worcester; MD: Baltimore; MI: Southfield; MN: Minneapolis; MO: Kansas City, St. Louis; NC: Charlotte, Greensboro; NH: Nashua; NJ: Cherry Hill, Wayne; NM: Albuquerque; NV: Las Vegas: NY: Buffalo, Latham, Melville, New York City, Newfield, Rochester, Syracuse; OH: Columbus, Dayton, Euclid; OK: Oklahoma City, Tulsa; OR; Portland; PA: Blue Bell, Carnegie; RI: Albion, Rumford; SC: Columbia; TN: Knoxville, Memphis; TX: Austin, Dallas, El Paso, Houston; UT: Salt Lake City; VA: Hampton, McLean, Norfolk, Richmond, Salem; WA: Kirkland; WI: Milwaukee; CANADA: Calgary, Alberta; Edmonton, Alberta; Richmond, B.C.; Ottawa, Ontario; Toronto, Ontario; St. Laurent (Montreal), Quebec.

INTERNATIONAL OFFICES: ARGENTINA: Buenos Aires; AUSTRALIA: Adelaide, Brisbane, Melbourne, Newcastle, Perth, Sydney; AUSTRIA: Vienna; BELGIUM: Brussels; BRAZIL: Sao Paulo; COLUMBIA: Bogata; COSTA RICA: San Jose; DENMARK: Copenhagen; ECUADOR: Quito; EGYPT: Cairo; FINLAND: Helsinki; FRANCE: Paris, Le Plessis Robinsons, Lyon; GERMANY: Dusseldorf, Eschborn, Filderstadt, Hamburg, Munich, Ratigen, GREECE: Athens; HONG KONG; IRAN: Teheran; ISRAEL: Tel Aviv; ITALY: Milan, Rome, JAPAN; Tokyo; KOREA: Seoul; KUWAIT: Kuwait City; LYBIA: Tripoli; MALAYSIA: Kuala Lumpur; MEXICO: Mexico City; NETHERLANDS: Rijswijk, NEW ZEALAND: Wellington; PERU: Lima; PHILLIPPINES: Makati, Metro, Manila; PORTUGAL: Lisbon; PUERTO RICO: Hato Rey; SAUDI ARABIA: Riyadh; SINGAPORE: Sri Lanka; SOUTH AFRICA: Johannesburg, Pretoria; SPAIN: Barcelona, Bilbao, San Sebastian, Valencia; SWEDEN: Gothenborg, Malmo, Stockholm; SWITZERLAND: Lausanne, Zurich; TAIWAN: Taipei; THAILAND: Bangkok; TURKEY: Ankara; UNITED KINGDOM: Birmingham; London; Manchester; Glasgow, Scotland; URUGUAY, Montevideo; USSR.

Data General

Data General Corporation, Westboro, Massachusetts 01581, (617) 366-8911

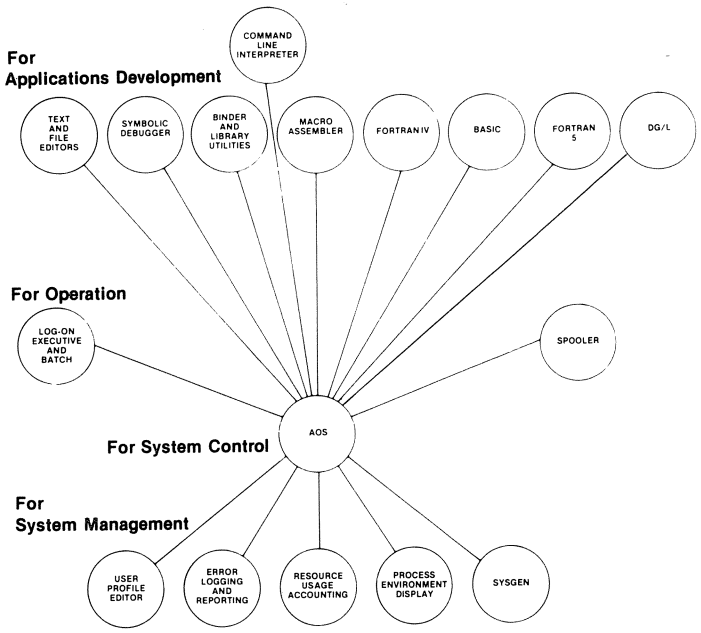
Data General

ADVANCED OPERATING SYSTEM (AOS)

PRODUCT BRIEF

FEATURES

- Intelligent, multiprogramming operating system that dynamically adapts to simultaneously control the three most common computer environments.
  - timesharing
  - multiple batch job streams
  - real-time
- Easy-to-use modules for both program development and production
- Dynamic management of all system resources based on user-established priorities and system-measured changes in resource usage
- Sophisticated multiprogramming techniques including:
  - Concurrent control of multiple system and user processes
  - Multitasking within processes
  - Intertask and interprocess communications
- Dynamic memory managment capabilities including:
  - Load-on-call access to routines and overlays
  - Generalized sharing of both code and data
  - Hardware-assisted memory protection
- Secure data management facilities including:
  - Multiple file-access levels controlled by user privileges
  - Hierarchial file directory structures
  - Device-independent input/output access procedures
  - Extensive communications facilities
- Powerful system management and operator modules including:
  - Log-on Executive with password protection
  - Batch facility with multiple job streams and operator queue management
  - System resource usage accounting
  - Output spooler with operator queue management
  - Hardware-error logging and on-line reporting with soft error retry and recovery
- User-oriented command line interpreter
- Interactive system generation
- Advanced support of the Data General ECLIPSE® computers and associated peripherals and communications capabilities



DESCRIPTION

Data General's Advanced Operating System (AOS) provides the intelligent, multiprogramming features of large-computer operating systems on the powerful ECLIPSE line of small computers. AOS has been explicitly designed to adapt dynamically to simultaneously control the three most common computer environments. Timesharing, real-time and multiple batch job stream operations are supported concurrently for production and/or program development. In addition, AOS intelligently performs program control, I/O and file management functions commonly performed by user programming on other computers.

AOS handles a wide variety of user environments. A system can simultaneously be a responsive multi-user timesharing system, an event-driven real-time system and a multiprogramming batch system.

The person authorized as AOS system manager intially assigns resources and privileges (e.g., execution priority and file access rights) uniquely to each user. Managers can let users change these priorities later or reserve the right to make changes themselves. Users may be permitted to access only a restricted set of files or they may be given wider latitude such as the ability to examine and modify other users' programs and data files.

The integrated AOS hardware/software protection scheme enforces these privileges and maintains integrity. And its memory management and multiprogramming capabilities allow multiple user activities to occur concurrently, making optimum use of system resources.

For application development, AOS provides high-level BASIC, FORTRAN IV, optimizing FORTRAN 5, Data General's system programming language (DG/L), a sophisticated Macro Assembler, and programming aids such as text editors and a symbolic debugger. For operation, AOS provides an Executive to control user log-on and ensure a convenient, secure multi-user environment where all users are protected from authorized interference. Its batch facilities control multiple simultaneous batch job streams. A prioritized queue is associated with the streams. Operators can interrogate the queue and select individual jobs for execution. Additional queue control includes hold/release, job deletion and entire queue deletion. The Spooler allows orderly usuage of output devices by independent users. For system managers, AOS includes a User Profile Editor to allocate resources and privileges, Resource Usuage Accounting software to account for resources utilized, a Process Environment Display to help manage the system and an Online Error Logging and Reporting capability.

At installation, the sytem manager generates an operating system tailored to the specific hardware configuration. From then on, AOS dynamically adapts itself and its modules to user demands. The type and behavior of the system's collective work-load dynamically determines the system's environment.

MULTIPROGRAMMING

AOS is a multiprogramming operating system oriented to processes, i.e., collections of program tasks sharing up to 64K bytes of address space. Each process is assigned a set of resources and privileges which determine how much time and memory the process can use. AOS controls a variable number of user and system processes independently and concurrently and protects the processes from each other's activities.

Users determine whether processes should be permanently resident in memory, pre-emptible (usually resident in memory but suitable for swapping), or swappable. The three types have specific applicability: resident processes can be used for real-time applications; pre-emptible processes suit less time-critical, event driven applications and swappable processes are ideal for timesharing and batch operations.

Initially, users assign resident and pre-emptible processes a priority ranging from 1 to 255 and swappable processes a priority of 1 to 3. More than one process can have the same priority. In addition, AOS imposes a hierarchy on processes to allow interprocess control. All processes existing at any one time are related in a "family-tree" structure and parent processes can block and unblock offspring. Processes can also assign offspring privileges (such as what data they can access).

In operation, AOS allocates CPU time to processes dynamically depending on their eligibility state. Processes are eligible if they have been allocated memory and are not waiting for some external event such as a console input/output or a message from another process. Among eligible processes, AOS executes the highest priority resident or pre-emptible process first. If none is eligible, it gives control to the highest priority swappable process.

In time-sharing operation, it is the swappable processes that are usually competing for system resources. AOS completely controls the swapping of these processes based on past behavior and process priority. For example, it grants more time to swappable processes that fully utilize their time slice but may assign them a lower priority.

MULTITASKING

AOS supports multiple tasks within any process. This provides efficient handling of multiple asynchronous events with a single user program.

When a process executes, AOS gives control to the task designated as first by the user. In a multitasking program, the first task initiates the other tasks. Then all can run in parallel or interact using task calls designed for execution control, communication and synchronization, and timing.

Execution control calls can suspend, make ready or terminate tasks based on the user-assigned priority or identification. Communication and synchronization calls can be used to send messages to other tasks, lock a process-wide resource, form "mailboxes" and queues, and distribute work among tasks. Task timing calls let users start tasks based on time-of-day or specific time intervals.

INTERTASK AND INTERPROCESS COMMUNICATION

AOS lets processes intercommunicate with multiple, point-to-point, free-format messages of any length. Processes send and receive messages using full-duplex communications ports and each process can have up to 127 such ports.

Process sending messages can continue activity without delay. If no receiver is waiting for the message, AOS can spool the message. Conversely, AOS can suspend a task that wishes to receive a message until another process sends the requested message.

MEMORY MANAGEMENT

AOS provides dynamic memory management assisted by the ECLIPSE-line Memory Allocation Protection (MAP) feature. Every process is dynamically allocated up to 32 2K-byte pages—the process logical address space—for inclusion of programs and data. AOS will choose any available physical pages for eligible processes. It never prevents or delays process loading because of lack of contiguous physical memory blocks or a lengthy restructuring of memory.

Memory pages are dynamically designated as either sharable or unsharable. Unshared pages contain information useable by only one process; shared pages can be used by any process with the proper access rights.

AOS divides physical memory into system and user logical address spaces. The specific memory space assigned to each user varies with the amount of user process activity.

Process type, swapping actions, and other calls interact to change the composition of user address space, which is further divided into shared and unshared areas. Each process can redefine the size of its shared and unshared areas dynamically. Shared areas are protected by access privileges (read, write, execute) granted to processes by users or other processes. When a process is swapped, its unshared areas are written to disc but its shared areas may remain for use by other processes.

AOS uses a least-recently-used (LRU) algorithm to ensure optimum use of shared pages and to reduce disc input/output. It coordinates memory page allocation and protection dynamically and automatically.

AOS overlay and shared routine load-on-call capabilities further increase memory efficiency. Shared routines are stored in libraries for access using only a name and number. Shared routines are hardware write-protected in their memory area and AOS automatically links the routines to the processes needing them at run time. Overlay functions are similar except that they are built into a program's logical address space before runtime.

ECLIPSE M/600 SYSTEM MANAGEMENT

On ECLIPSE M/600 systems, AOS efficiently manages the Job Processor, Demand Paged Storage Facility, Burst Multiplexor Channel and the Input/Output Processor. The Job Processor handles the scheduling and execution of interactive and batch users' jobs. The Demand Paged Storage Facility allows a minimum portion of a user's program to be brought into main memory initially with all other portions being brought in on demand. This feature lets a greater number of users access the computer. AOS works closely with this facility to load new pages of user programs as they are needed and swap altered pages back to disc when necessary. Memory is managed on a least-recently-used (LRU) basis.

The Burst Multiplexor Channel interfaces to and enhances the operation of high speed peripheral devices. Low-speed character-oriented devices are processed by the independent Input/Output Processor, freeing the Job Processor for high speed execution of system and user functions. AOS coordinates the operation of these M/600 features.

FILE MANAGEMENT

AOS provides data management capabilities including complete file protection by access, device-independent I/O access, and hierarchial file directories to catalog programs and data.

Like processes, all file directories are related in a "family-tree" structure. The initial parent is termed the system root. The system root directory contains entries for other directories and certain files used by AOS.

AOS has a number of predefined directory entry types such as program file, peripheral device, interprocessor communication file, and other system types. Users can define up to 128 additional directory entry types. Each entry contains a name and other information unique to that type of entry.

Every process has a working directory and a search list of other directories to consult if a process cannot find a named entry in the processes' working directory.

FILE ACCESS CONTROL

AOS provides a secure protection scheme that prevents unauthorized or accidental use or alteration of data accessed through the directory structure. Privileges include owner, write, append, read and execute access. These privileges differ slightly depending on whether an entry is in fact another directory or a data file. The owner can change privileges during the life an an entry. AOS maintains an access control list for each directory and data file. It includes the users that can access the files as well as the privileges that access allows.

This access control mechanism provides sophisticated protection of code and data in the system while simultaneously simplifying procedures for the individual user.

DEVICE-INDEPENDENT INPUT/OUTPUT

AOS provides a flexible system for accessing files on ECLIPSE-line peripheral devices. To execute an I/O transfer to any device, users simply open the file, read or write file data, and close the file. Users can read/write data by blocks or by logical groupings called records. This method is used to transfer data to and from all devices including remote devices.

Various Data General disc units can be mixed on any system. At installation, the user specifies one or more physical disc drives as a logical disc unit.

Users can access magnetic tape using block or record I/O methods. Access control applies to the entire tape. AOS implements ANSI standard levels 1, 2 and 3 and IBM levels 1 and 2 for labeled magnetic tape. User programs can read and write tapes with these labels for convenient data interchange.

The Multiprocessor Communications Adapter (MCA) is treated as a multi-file device. MCA block transfers use a protocol compatible with Data General's Real-time Disc Operating System (RDOS) and Real-time Operating System (RTOS).

Utilization of synchronous and asynchronous communications lines with or without Data General's DCU/50 is completely transparent to the user.

For all file I/O, dynamic, fixed-length, data-sensitive and variable-length records are supported. Users can specify the record type when creating a file or when performing the input/output.

Block data transfers are supported on disc units, magnetic tape units and multiprocessor communications adapters. Disc unit blocks are 512 bytes long; magnetic tape unit blocks vary in size and are separated by interrecord gaps; multiprocessor communications blocks can vary in length up to 8192 bytes.

CONCURRENT DIAGNOSTIC CAPABILITIES

AOS supports diagnostic programs running concurrently with other AOS processing. These programs exercise the Central Processing Unit, its memory and options as well as peripheral devices. This capability lets a user verify system device operations while continuing all normal AOS activities.

COMMAND LINE INTERPRETER

The AOS Command Line Interpreter (CLI) lets users perform file and process maintenance at the console. CLI is also used to invoke execution of other AOS modules or user programs.

CLI commands are used to execute programs, execute functions built into the CLI itself, and expanduser-or system-defined macros.

Specifically, some of the operations CLI performs are: file creation, maintenance and backup; directory creation and maintenance;spooling control, and batch job submission.

One of CLI's great advantages is its flexibility. Some CLI functions do little more than execute one system call (such as DELETE "filename", while others (such as DUMP) perform very complex operations. Depending on user needs, they can learn CLI commands ranging from a few simple ones to the large number available for more complex processing.

All CLI commands can be abbreviated. For example, the full name of DIRECTORY can be shortened to DIRECT or DIR or any letters that don't conflict with another command.

A HELP capability is available to supply users more information on any CLI command. For example, users can type, "HELP DIRECTORY" to receive a full description of the DIRECTORY command.

A macro facility enables users to combine and execute a series of commands with a single name while passing variable parameters.

CLI macro features include conditional command execution, compound commands, template expansion, variables and error handling. CLI macros can call themselves or other CLI macros.

UTILITIES

A wide variety of system utilities provide program development and system management tools. Program development tools include:

- Symbolic Text Editor, which lets users conveniently edit program source and data files
- Binder, which links object files to produce an executable program file and overlay file
- Sharable Symbolic Debugger, which lets users examine memory, set break points and test execution interactively
- Shared Library Builder program, which easily creates, edits and maintains shared routine libraries
- Symbolic File Editor, which allows modification of program files

With its Spooler, AOS ensures that multiple users sending output to line printers directly or via asynchronous communications lines receive service in an orderly manner. The Spooler provides forms control, flush and print and restart capabilities. The Spooling mechanism diverts output to a temporary disc file. Each printer on the system has entries in the spool queue containing the names of files intended for it.

A User Profile Editor utility allows system managers to specify user privileges and Resource Usage Accounting software maintains user-level accounting information. In addition, the Executive validates a log-on user or batch job against the user's profile. Then it creates a process for each, according to the privileges assigned and monitors users' activities.

USER PROGRAMMING LANGUAGES

AOS user programming languages fulfill a variety of commercial, scientific and computational applications. FORTRAN IV comes with a series of ISA extensions. FORTRAN 5 contains the same extensions plus features which globally optimize code for faster execution. BASIC is suitable for educational and scientific applications. Data General's system programming language (DG/L) is an excellent system development tool. And the AOS Macro Assembler provides the most precise control of ECLIPSE hardware; it includes macro facilities with extensive conditional assembly that allows users to define a complex instruction series as a single instruction.

SYNCHRONOUS COMMUNICATIONS

AOS supports the IBM BISYNC synchronous communications line protocol. The Macro Assembler provides a direct access to point-to-point or multidrop communications lines, with or without DCU/50 assistance. Also, AOS supports two remote batch emulation packages. These enable AOS systems to function as a host for other Data General computer systems, or to communicate with IBM 360/370 compatible systems. The RJE80 utility emulates IBM 2780 and 3780 remote job entry protocol. And the HASP II utility emulates an IBM HASP II remote job entry workstation (IBM 360/20).

SYSTEM GENERATION

The person designated as system manager can generate an AOS system tailored to the hardware. Very flexible device specification (modem, baud rate, cursor controls, tab simulation, etc.) is permitted. These specifications may be dynamically changed during system operation.

In addition, multiple operating systems can be generated and stored on disc for easy access. AOS offers the system manager considerable flexibility in specifying these systems. The entire generation process is interactive.

SUPPORTED HARDWARE

AOS operates on Data General ECLIPSE computers with at least 192 bytes of memory.

AOS supports a wide variety of peripherals, including fixed- and moving head discs and diskettes, magnetic tape units, terminal printers, video displays, card readers, line printers, digital plotters, paper tape readers and punches.

AOS also supports Data General's Communication System (DG/CS), Data Control Unit/50 and the Multiprocessor Communications Adapter.

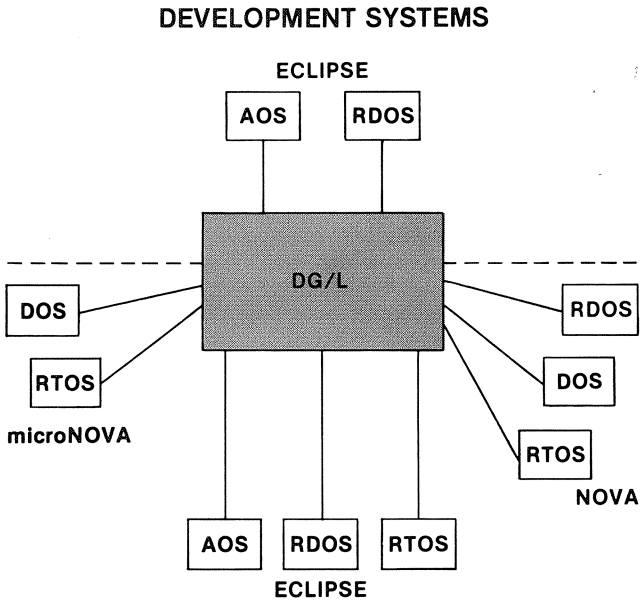
The DG/Disc Fixed Head Subsystem enhances AOS performance in a multi-user swapping environment. The advanced controller features, fast access and high-speed transfer capabilties of this disc can provide higher system throughput. AOS coordinates the ECLIPSE M/600 Demand-Paged Storage facility with fixed-head disc subsystems to augment overall system processing.

Data General

Data General Corporation, Westboro, Massachusetts 01581, (617) 366-8911

FEATURES

- Compiler executes under Data General's Advanced Operating System (AOS) and mapped Real-time Disc Operating System (RDOS) on ECLIPSE® computers
- Object programs execute on Data General ECLIPSE, NOVA®, and microNOVA computers under AOS, RDOS, DOS, and RTOS operating systems
- ALGOL-like syntax for structured programming
- Globally optimizing compiler for re-entrant and recursive code
- Memory management techniques, including ALLOCATE/FREE statements and cache memory handlers
- Direct address manipulation and based variables
- Character and bit string manipulation and arithmetic
- Single- and double-precision integer arithmetic
- Single- and double-precision real arithmetic
- Comprehensive mathematical function library
- Multitasking runtime environment



DESCRIPTION

Data General's DG/L is an ALGOL-derivative structured programming language designed for a wide range of applications. DG/L permits development on ECLIPSE computers for execution across Data General's ECLIPSE, NOVA, and microNOVA product lines. DG/L's comprehensive operating system interface, multitasking runtime environment, and memory management features make it especially useful for multiterminal applications. Its arithmetic and mathematical library functions make it ideal for computation tasks; and its string handling and string arithmetic features enable commercial applications. DG/L is an ideal development tool for systems-level software, such as compilers, assemblers, sort/merges, and other utilities. The DG/L compiler globally optimizes user source code, and the code generated is re-entrant and recursive. Data General's DG/L is intended for use by sophisticated software product OEMs, data processing organizations and academic users.

OPERATING SYSTEMS

The DG/L compiler runs on Data General ECLIPSE computers under the Advanced Operating System (AOS) and mapped Real-time Disc Operating System (RDOS). DG/L object programs run on Data General ECLIPSE, NOVA, and microNOVA computers under the AOS, RDOS, Real-Time Operating System (RTOS) and Disc Operating System (DOS).

DG/L's source language syntax lets users code in one form, regardless of the operating system being used. The compiler and runtime library provide the proper object-time interface.

With an AOS program development system, users can write, compile and debug code under AOS, create the RDOS, RTOS or DOS executable file, and use magnetic tape or a diskette for transporting to the object computer. Compile-time options are used to control code generation for AOS, RDOS, RTOS or DOS.

MEMORY MANAGEMENT

DG/L provides flexible tools for dynamic memory management, including local and global ALLOCATE/FREE statements and cache memory management. The runtime call ALLOCATE assigns words from free space left in a partition after program code and stack space allocation. A FREE call releases the memory. In a multitasking environment, competing tasks may use these features to share a large block of memory that each individually needs for brief periods.

The DG/L cache memory manager lets programmers deal with data structures that are larger than the program's address space. Programs may access up to 64K elements in each cache memory structure. Each element is a user-defined number of words. One or more elements form a node, which is accessed by runtime calls as though it were an element of a large array. The data is actually stored on disc as a file. The memory areas used by the cache memory management feature are managed with a least-recently-used (LRU) algorithm to increase program efficiency and speed data access.

DG/L's automatic variable scoping further improves memory use. Storage areas required by variables local to a given procedure are dynamically allocated and de-allocated as the procedure is entered and exited.

ADDRESS MANIPULATION

DG/L lets programmers manipulate storage addresses directly to create parameter packets, linked lists, and other systems programming constructs. DG/L also includes a based variable capability. This is a variable declaration that requires no storage, and uses an address pointer to access a storage template at runtime.

STRING FUNCTIONS

DG/L's extensive string management functions make it well suited for commercial applications. The "substring" and "index" runtime routines allow concatenation and string parsing. DG/L also supports string variables in arithmetic operations with automatic decimal point alignment and arbitrarily large precision.

ARITHMETIC

DG/L supports single- and double-precision integer arithmetic. Single- and double-precision real arithmetic is optionally supported on computers with a hardware floating point feature.

MULTITASKING

DG/L has a full multitasking runtime environment. It can access all the task creation, control, and synchronization features of the Data General operating system being used. Task creation calls define the stack size for the created task, and each task can use ALLOCATE/FREE storage management and XMT/REC synchronization to further manage memory resources. This makes DG/L an ideal development tool for RDOS, RTOS, or DOS multiterminal applications.

HARDWARE REQUIREMENTS

The minimum hardware requirement for a DG/L compiler and runtime routines operating under RDOS is an ECLIPSE system licensed for RDOS, with at least 96K bytes of read/write memory. For a compiler and runtime routines operating under AOS, users need an ECLIPSE system licensed to run AOS, with a minimum of 192K bytes of read/write memory. Runtime routines licensed for NOVA or ECLIPSE systems require 32K bytes of memory and a real-time clock.

SALES AND SERVICE

NORTH AMERICAN OFFICES: Westboro, Massachusetts, 01581, (617) 366-8911 Headquarters. And AL: Birmingham; AZ: Phoenix, Tucson; CA: El Segundo, Palo Alto, Paramount, Sacramento, San Diego, San Francisco, Santa Ana, Santa Barbara, Van Nuys, Woodland Hills; CO: Engelwood; CT: North Branford; FL: Ft. Lauderdale, Orlando, Tampa; GA Atlanta; ID: Boise; IL: Peoria, Schaumburg; IN: Indianapolis; KY: Louisville; LA: Baton Rouge; MA: Cambridge, Springfield, Wellesley, Worcester; MD: Baltimore; MI: Southfield; MN: Minneapolis; MO: Kansas City, St. Louis; NC: Charlotte, Greensboro; NH: Nashua; NJ: Cherry Hill, Wayne; NM: Albuquerque; NV: Las Vegas: NY: Buffalo, Latham, Melville, New York City, Newfield, Rochester, Syracuse; OH: Columbus, Dayton, Euclid; OK: Oklahoma City, Tulsa; OR; Portland; PA: Blue Bell, Carnegie; RI: Albion, Rumford; SC: Columbia; TN: Knoxville, Memphis; TX: Austin, Dallas, El Paso, Houston; UT: Salt Lake City; VA: Hampton, McLean, Norfolk, Richmond, Salem; WA: Kirkland; WI: Milwaukee; CANADA: Calgary, Alberta; Edmonton, Alberta; Richmond, B.C.; Ottawa, Ontario; Toronto, Ontario; St. Laurent (Montreal), Quebec.

INTERNATIONAL OFFICES: ARGENTINA: Buenos Aires; AUSTRALIA: Adelaide, Brisbane, Melbourne, Newcastle, Perth, Sydney; AUSTRIA: Vienna; BELGIUM: Brussels; BRAZIL: Sao Paulo; COLUMBIA: Bogata; COSTA RICA: San Jose; DENMARK: Copenhagen; ECUADOR: Quito; EGYPT: Cairo; FINLAND: Helsinki; FRANCE: Paris, Le Plessis Robinsons, Lyon; GERMANY: Dusseldorf, Eschborn, Filderstadt, Hamburg, Munich, Ratigen, GREECE: Athens; HONG KONG; IRAN: Teheran; ISRAEL: Tel Aviv; ITALY: Milan, Rome, JAPAN; Tokyo; KOREA: Seoul; KUWAIT: Kuwait City; LYBIA: Tripoli; MALAYSIA: Kuala Lumpur; MEXICO: Mexico City; NETHERLANDS: Rijswijk, NEW ZEALAND: Wellington; PERU: Lima; PHILLIPPINES: Makati, Metro, Manila; PORTUGAL: Lisbon; PUERTO RICO: Hato Rey; SAUDI ARABIA: Riyadh; SINGAPORE: Sri Lanka; SOUTH AFRICA: Johannesburg, Pretoria; SPAIN: Barcelona, Bilbao, San Sebastian, Valencia; SWEDEN: Gothenborg, Malmo, Stockholm; SWITZERLAND: Lausanne, Zurich; TAIWAN: Taipei; THAILAND: Bangkok; TURKEY: Ankara; UNITED KINGDOM: Birmingham; London; Manchester; Glasgow, Scotland; URUGUAY, Montevideo; USSR.

Data General

Data General Corporation, Westboro, Massachusetts 01581, (617) 366-8911