Control Data 6600. Glenview, Ill.: Scott Foresman.

1971. Bell, C. G., and A. Newell. *Computer Structures; Readings and Examples.* New York: McGraw-Hill.

1972. Hintz, R. G., and D. P. Tate. "Control Data STAR-100 Processor Design," Compcon 72, *IEEE Computer Society Conference Proceedings* (September), pp. 1–4.

D. J. KUCK AND D. H. LAWRIE

SUPERVISOR CALL

For articles on related subjects *see* INTERRUPT; MULTIPROGRAMMING; OPERATING SYSTEMS; PRIVILEGED INSTRUCTION; and SUPERVISOR STATE.

A typical operating system has a set of system programs collectively known as the "supervisor," whose function it is to provide services for and to supervise the running of a number of user programs. Control goes to the supervisor every time the normal flow of processing is interrupted by a change of state in the system.

The purpose of a *supervisor call* is to provide a mechanism whereby a program can interrupt the normal flow of processing and ask the supervisor to perform a function for the program that the program either cannot or is not permitted to perform for itself.

The most typical supervisor calls have to do with input and output. In a multiprogramming system it is essential to have system control of input/output devices, especially those devices shared by a number of programs.

Most computers that were designed for multiprogramming systems have a supervisory mode of operation and hardware interlocks that prevent certain supervisory operations from taking place except when the computer is operating in supervisory mode. This may be handled by means of special privileged instructions that can be executed only in supervisory mode, or only in some other way.

In the IBM 360/370 systems, for example, a supervisor call is made through the execution of an instruction whose effect is to create an interrupt. The instruction is 2 bytes long. The first byte is the supervisor-call instruction code, and the second byte describes the nature of the supervisor call. This second byte goes into a special register which is used in connection with all interrupts to transmit information to the system as to the status of that particular interrupt.

The interrupt now proceeds like any other interrupt. It stores the status of the computer (the old program status word) and loads a new status that gives control to a resident supervisory routine, which operates in supervisor mode and whose function is the handling of supervisor calls. This routine then analyzes the second byte of the supervisor-call instruction and determines the nature of the call.

It is, of course, possible—and usually essential—that additional information is passed to the supervisory routine as a result of the supervisor call. This information may be in special registers (general registers) or in an area of memory pointed to by a special register.

The supervisor may have resident routines for handling certain classes of supervisor calls, and may have available areas of central memory (transient areas) into which overlays can be loaded for the handling of less frequent supervisor calls. Fast response to supervisor calls is usually an important factor in system performance, and systems that have large amounts of central memory can often improve their responsiveness by increasing the number of resident supervisor-call routines.

S. ROSEN

SWAC

For articles on related subjects *see* DIGITAL COMPUTERS: Early; and SEAC.

SWAC (National Bureau of Standards Western Automatic Computer) was dedicated in August 1950, and at the time of its dedication was the fastest computer in existence. It was begun in January 1949 at the National Bureau of Standard's field station, the Institute for Numerical Analysis at the University of California at Los Angeles, and was designed and constructed under the direction of the author. Originally named the ZEPHYR, due to its modest-sized budget and staff as contrasted with much larger projects being carried on elsewhere, it was later renamed the SWAC.

The SWAC was a parallel computer using Williams' tube (cathode-ray tube, or CRT) memory. The memory cycle was 16 μs consisting of an 8 μs action cycle and an 8 μs restore cycle (where some

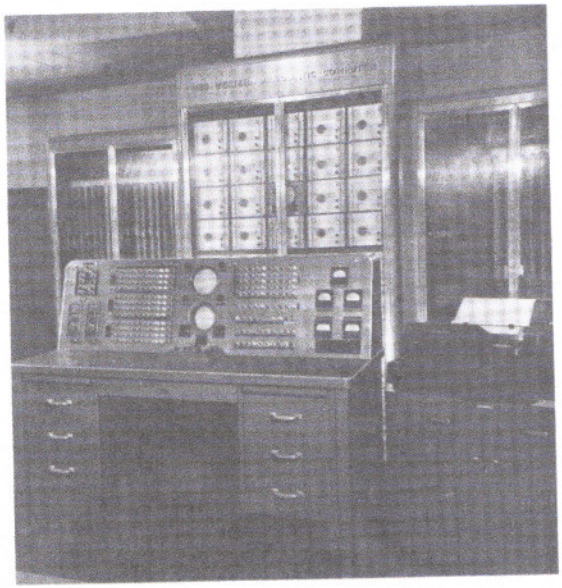

Fig. 1. The SWAC.

other memory location was restored). An addition of 37-bit operands occurred in 64 μs, and multiplication occurred in 384 μs. Due to technical difficulties with Williams' tube storage, the memory was never increased beyond 256 words. A 4,096-word magnetic drum was added to the system with coordinated addressing so that block transfers of 32 words between the two memories occurred with no latency.

Initial input and output was by typewriter and punched paper tape. These were soon replaced by a card reader (240 cards per minute) and a card punch (80 cards per minute). The SWAC used a four-address command structure. A floating-point interpretive system named SWACPEC was developed, which made it much easier for users to write programs.

In 1953 the SWAC was producing about 53 hr of useful computing time per week. SWAC was used in a research computing environment, and therefore many of the problems tended to be quite large. Solution times from 177 to 453 hr are reported by Huskey et al. (1953). Some of the early problems included the search for Mersenne primes, the Fourier synthesis of X-ray diffraction patterns of crystals, the solution of systems of linear equations, and problems in differential equations.

When the National Bureau of Standards ceased to support the Institute for Numerical Analysis, the SWAC was transferred to the University and moved to the Engineering Building at UCLA. There it continued in useful operation until December 1967. Parts of the SWAC are now on exhibit in the Museum of Science and Industry in Los Angeles.

References

1951. Huskey, H. D. "Semiautomatic Instruction on the Zephyr," *Proceedings of a Second Symposium on Large-Scale Digital Computing Machinery*. Cambridge, Mass.: Harvard University Press, pp. 83–90.

1953. Huskey, H. D., R. Thorensen, B. F. Ambrosio, and E. C. Yowell. "The SWAC—Design Features and Operating Experience," *Proceedings of the I.R.E.*, Vol. 41, No. 10 (October), pp. 1294–1299.

H. D. Huskey

SWAPPING

For articles on related subjects *see* Memory: Auxiliary; Scheduling Algorithm; Time Sharing; and Time Slice.
For articles on related terms *see* Algorithm; and Working Set.

Swapping is a process found most frequently in time-shared computer operating systems. It is used to move programs between primary storage (usually core memory, but integrated circuit memories have recently become common) and secondary storage (drums or disk files). Swapping is necessary to maximize the efficient use of valuable primary storage where programs must be located in order to be executed, but which cost an order of magnitude more than secondary storage.

When the operating system locates a program in a "waiting" state—i.e., one that can no longer use the CPU normally because it is waiting to complete an I/O operation—it *swaps* the program to secondary storage, replacing it with another program that is ready to use the CPU. In time-sharing systems, the longest waiting state occurs as the program awaits input from the relatively slow typewriter, Teletype, or similar terminal.

The time-sharing systems of the early 1960s at M.I.T., The RAND Corporation, Systems Development Corporation, and Dartmouth College contain examples of the swapping process. In these