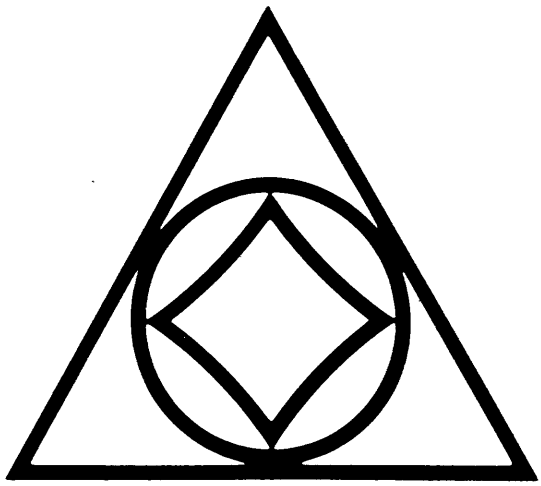


AFIPS

CONFERENCE PROCEEDINGS

VOLUME 29

1966

FALL JOINT COMPUTER CONFERENCE

NOVEMBER 7–10
SAN FRANCISCO, CALIFORNIA

SCIENTIFIC DATA SYSTEMS

SDS
SCIENTIFIC DATA SYSTEMS

THE SDS SIGMA 7: A REAL-TIME TIME-SHARING COMPUTER

Myron J. Mendelson and A. W. England

Scientific Data Systems, Santa Monica, California

INTRODUCTION

The SDS SIGMA 7 Computer system (Fig. 1) is unique among new computer designs in that it is the only system which has seriously considered and solved the problem of achieving true real-time response hardware and software capability while operating in a multiprogramming, multiprocessing, space-sharing, and time-sharing environment. This paper presents an overview of the system's architecture and describes in some detail those of its features which provide its unique capabilities.

The paper is divided into two major portions. The first part presents a succinct description of the architecture of the system. Its purpose is to acquaint the reader with the fundamental characteristics of SIGMA 7 and to provide a meaningful framework for the second section. The second section delineates seven major problems which were considered critical in the design of SIGMA 7 and presents the details of their solution.

DEFINITIONS

Multiusage

We will use the generic term "multiusage" to cover the spectrum of multiprogramming, multiprocessing, space-sharing, and time-sharing operations. These, together with the term "real time," will have the following meanings:

Real-Time Operation. A true real-time operation is one in which the response time requirements of the system are imposed by the time sensitive de-

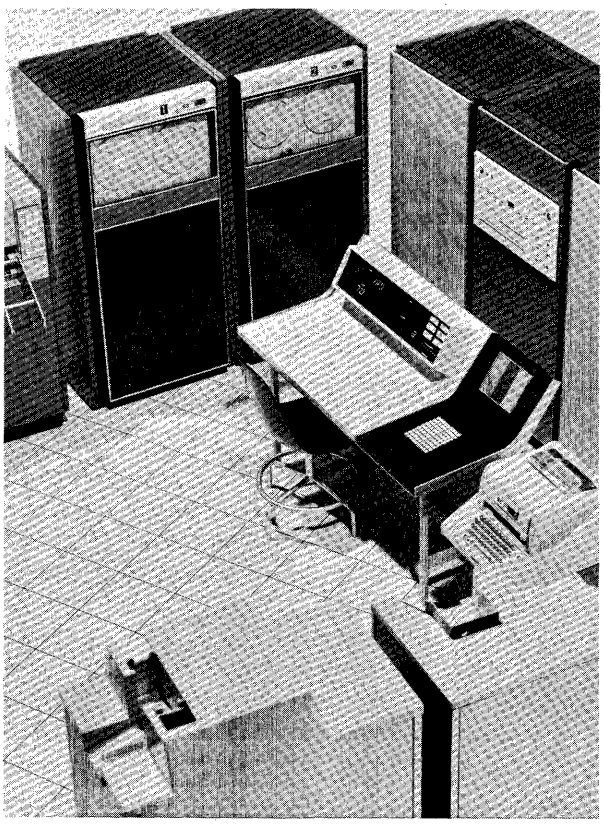


Figure 1. SIGMA 7 computer system.

mands of events external to the computer and its conventional peripheral equipment. Failure to meet this response time requirement results in true failure of the real-time system, not just degraded performance. The responsiveness of such a system is measured by the time interval between the arrival of an interrupt trigger signal and the execution of the first *useful* instruction in response to it. In the extreme case the maximum acceptable length for this interval may be measured in microseconds.

Multiprogramming. Multiprogramming is the concurrent operation of two or more independent programs in a single computing system, control being switched among programs through the actions of some central control program.

Multiprocessing. Multiprocessing is the simultaneous execution of one or more programs in a single computing system containing two or more processors, preferably sharing a common memory pool.

Space Sharing. Space sharing is the simultaneous residency in a common central memory of a number of independent (and perhaps concurrently operating) programs.

Time Sharing. Time sharing is a special case of multiprogramming in which a multiplicity of separate users have on-line, interactive use of a common system. It should be noted that neither multiprogramming nor time sharing imply space sharing but that space sharing is an essential ingredient in achieving true efficiency in these operations.

SYSTEM ORGANIZATION

Introduction

The following brief description of the SIGMA 7 system is presented in order to provide a meaningful framework within which to describe the specialized features which provide the system with a unique capability to meet its design goals. The SIGMA 7 is a modularly organized system which is configured out of a combination of Central Processing Units (CPU) (which contain Priority Interrupt Systems), Memory Modules, Fast Memory Units, Multiplexing Input/Output Processors (MIOP), Selector Input/Output Processors (SIOP), peripheral equipment Device Controllers (DC), peripheral Devices (D), and specialized real-time interfaces such as Analog-to-Digital Converters, Digital-to-Analog Converters, and Multiplexers (Fig. 2). This paper will concentrate on the characteristics and structures of the CPU, IOP's and memory systems and their organization to meet the requirements of a broad range of operating environments.

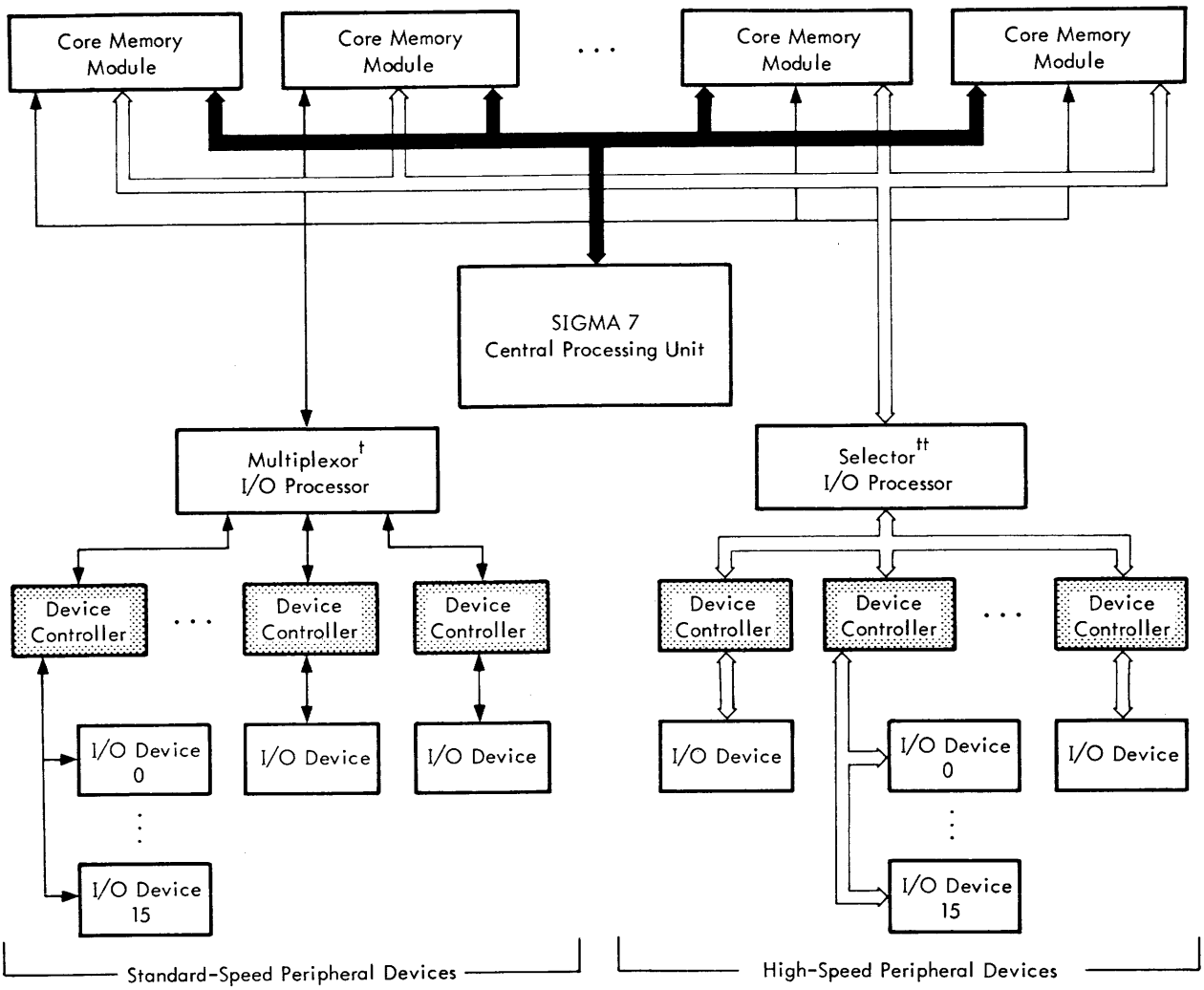
Memory Organization

Core Memory Modules. The SIGMA 7 core memory is a 32 bit plus parity bit, word organized, 850 nanosecond cycle time unit which is available in module sizes of 4K, 8K, 12K, and 16K words (K=1024). The system architecture permits the inclusion and direct addressing of any size memory which can be configured within eight memory modules. This permits the structuring of 32 different memory sizes ranging from 4K words (16K bytes) to 128K words (512K bytes). Although the memory is word organized and word parity checked it is capable of altering less than a full word on a write operation. From 1 to 3 bytes may be written without altering the remaining bytes.

Multiple Ports. The processor/memory system complex is a bus-organized asynchronously operating system with each processor having its own private bus. The standard memory module is equipped with two independent access paths (called ports) and an optional third port may be added. Subsequent to the addition of a third port a memory port expander provides for the four way expansion of any *single* port so that a maximum of six independent buses may be connected to any memory module. The ports have a fixed priority relationship with respect to each other so that access request conflicts are automatically resolved.

Asynchronous, Overlapped Operation and Memory Interleaving. Memory operations may be initiated at any time and are not synchronized to any central clocking source. Memory operations are self-sustaining so that processor release occurs upon data acceptance (by the memory) on a write operation, and processor "go-ahead" occurs upon data availability on a read operation. This permits maximum utilization of CPU time and the overlapping of memory cycles with respect to a single processor or multiple processors in multiple-module memory configurations. To insure memory overlapping under any circumstances, address interleaving among several memory modules is provided on a two-way or four-way basis.

Fast Memory Units. An integrated circuit, nondestructively read fast memory unit with a read cy-



† Multiplexor IOP allows up to 32 devices (one per device controller) to operate simultaneously with a combined transfer rate of 500,000 bytes per second.

†† Selector IOP allows one device at a time to operate at a transfer rate of up to 3 million bytes per second. A selector IOP may service up to 32 high-speed devices, and two selector IOPs may share a single memory bus.

Figure 2. A typical SIGMA 7 system.

cle time of 60 nanoseconds and a write cycle time of 90 nanoseconds is used to implement a number of special functions within the SIGMA 7 system. The basic building block fast memory module provides 16 bytes of operating storage. Four such modules are combined to provide 16 words of scratchpad memory which serves as register storage for the CPU. Other combinations of this single module type are used for the implementation of memory protection systems, fragmentation and dynamic program relocation techniques, IOP channel control functions, and device buffering systems.

Every instruction makes one or more references to a set of sixteen registers. These registers are stored in a sixteen word fast memory unit which is designated as a "register block."

In general, the SIGMA 7 register block can be used to provide:

1. 16 separate single precision arithmetic registers for fixed point word operations or short floating point operations.
2. 16 separate double precision arithmetic registers for fixed point halfword operations.
3. 8 separate double precision arithmetic registers for fixed point double precision operations or long floating point operations.

4. 7 separate index registers.
5. A decimal accumulator with a maximum capacity of 31 digits plus sign.
6. A significance position marking register for the EDIT instruction.
7. Control registers for byte string instruction implementation.

A unique design feature of the SIGMA 7 is that it may contain up to 32 blocks of registers. A 5-bit register pointer designates which of the 32 is currently active. The provision of multiple blocks makes it possible to preserve one register set and establish a new one within the 6 microsecond execution period of a single environment preserving and switching instruction.

Central Processing Unit

The CPU (Fig. 3) is a 32-bit, word-oriented, parallel-operating unit employing multiple registers in its instruction implementation. Its extensive instruction set provides for operations on 8 bit-bytes, 16-bit halfwords, 20-bit immediate operands, 32-bit words, and 64-bit doublewords.

Instruction Format. SIGMA 7 provides 106 major instructions, many of which have multiple modes of operation, all contained within a single instruction format. The Basic instruction is 32 bits in length and has the structure shown in Fig. 4. For a special class of immediate operand instructions the X and M fields are combined into a single 20-bit value which is sign extended and used immediately for computation with no further reference to memory for an operand.

Direct Memory Word Addressing. The 17-bit word address field in the primary instruction word permits the direct addressing of the maximum sized 128K word memory system. A memory address in the range 0-15 is used to designate the correspondingly numbered register and does not result in access to core memory. Hence, the full power of the instruction set may be applied to register-to-register operations as well as to register-and-memory operations.

Indirect Addressing. Indirect addressing is included for all instructions except those of the immediate operand class. If both indirect addressing and indexing are invoked, the indirect address operation is executed prior to the indexing operation.

Indexing. The indexing operation employed in SIGMA 7 is unique. The indexing operation assumes that a list of either bytes, halfwords, words, or doublewords is stored beginning at the word address contained in the primary instruction word. If the designated index register is considered to contain the value K, the indexing operation, under control of the operation code (which establishes the operand length), produces the address of the byte, halfword, word, or doubleword displaced K units from this word location. Thus, the same index register may be used to locate the K^{th} operand of a list *independent of the operand length* (Fig. 5).

Instruction Set. The SIGMA 7 instruction set is comprehensive. It includes fixed point load, store, arithmetic, logical, and comparison operations for bytes, halfwords, 20-bit immediate operands, words, and doublewords. Optional floating point instructions provide full floating point arithmetic capability for both short (32-bit) and long (64-bit) formats. An optional set of decimal instructions includes full decimal arithmetic capability, plus Pack, Unpack, and Edit. Standard instructions are provided for manipulating byte strings up to 255 bytes in length. Single instructions are provided for moving a string, for comparing two strings, for the translation of a string from one character code to another, and for the scanning of a string for a specified set of characteristics. Push-down stack instructions provide for the efficient manipulation (including automatic stack limit checking) of arbitrary size stacks in core memory. Two generalized conversion instructions provide for the high speed conversion between any weighted binary information representation used external to the computer and its equivalent internal binary representation. Read Direct and Write Direct instructions provide for the direct communication between the CPU and external equipment without the use of an I/O channel. A comprehensive set of branch and system control instructions complete the instruction repertoire.

Typical instruction execution times (including indexing and mapping and excluding any memory overlap) are:

Fixed Point Add/Subtract	2.26 microseconds
Fixed Point Multiply	4.9 microseconds
Fixed Point Divide	12.5 microseconds
Floating Point Add/Subtract (short)	3.9 microseconds

Floating Point Add/Subtract (long)	4.5 microseconds
Floating Point Multiply (short)	5.4 microseconds
Floating Point Multiply (long)	8.0 microseconds
Floating Point Divide (short)	12.3 microseconds
Floating Point Divide (long)	24.5 microseconds

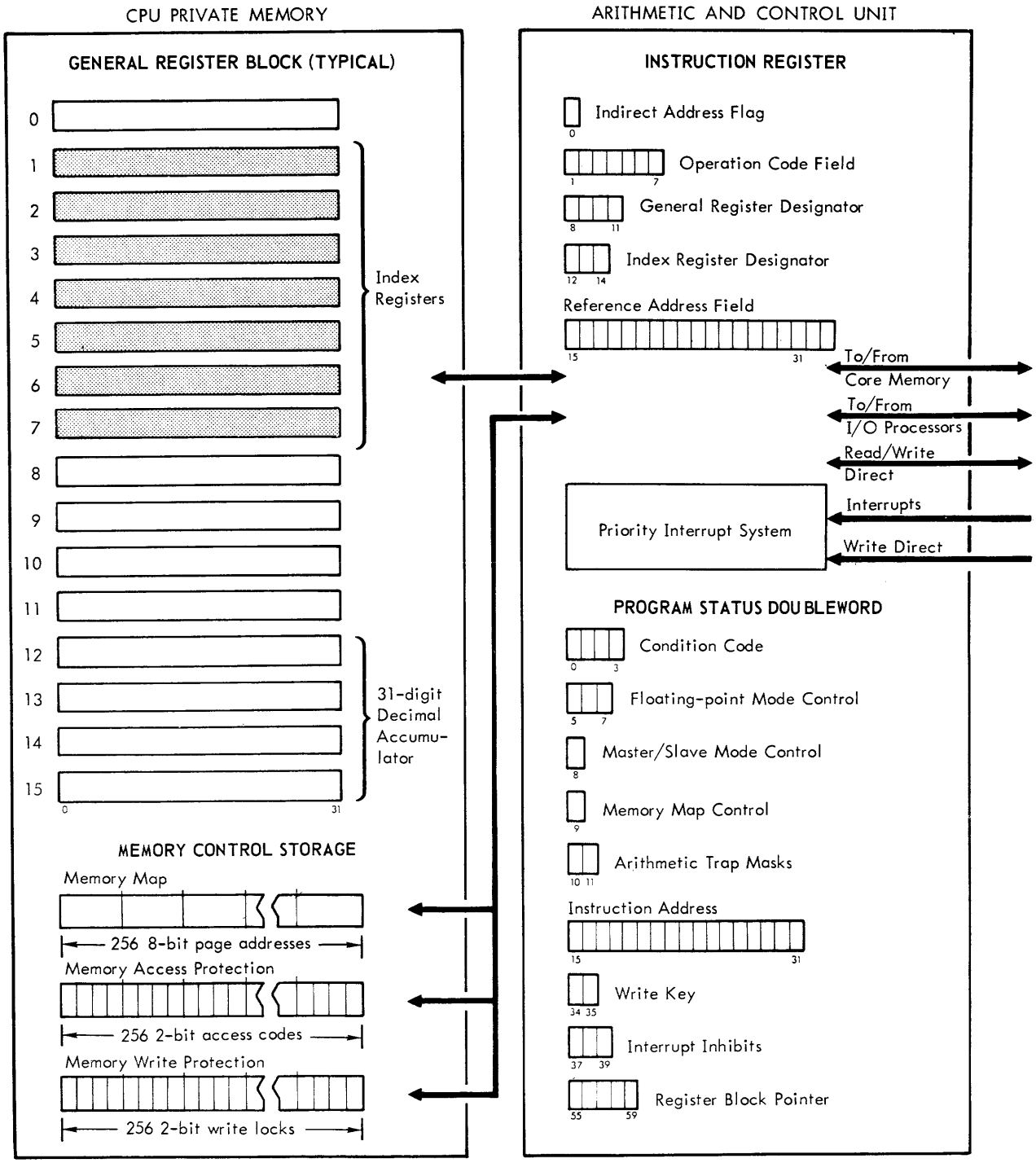


Figure 3. SIGMA 7 central processing unit.

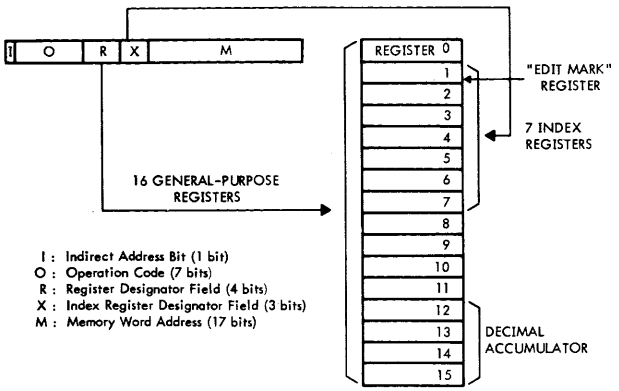


Figure 4. Register and instruction format.

Priority Interrupt System. The SIGMA 7 is equipped with the most powerful and flexible priority interrupt system currently available. Since this system constitutes one of the major elements contributing to the real-time responsiveness of the SIGMA 7 its description will be deferred to a later point in this paper.

Input/Output Organization

Multiplexing and Selector Type Input/Output Processors. The Multiplexer Input/Output processor (MIOP) is designed to service a large number of slow to medium speed peripheral devices simultaneously. A single MIOP can provide concurrent service to as many as 32 devices having a total bandwidth of approximately 500,000 8-bit bytes per second. A single Selector Input/Output Processor (SIOP), with the capability of operating at rates up to 3 million bytes per second is designed to service any one of as many as 128 high speed devices which may be attached to it. SIOPs may have private buses or two may share a common bus. As many as eight IOP's may be attached to a single CPU. Each operates independently of the CPU under control of a stored program which is held in core memory. The CPU activates and monitors the I/O operations through the use of a set of five I/O instructions. Once activated the sequencing of the stored I/O

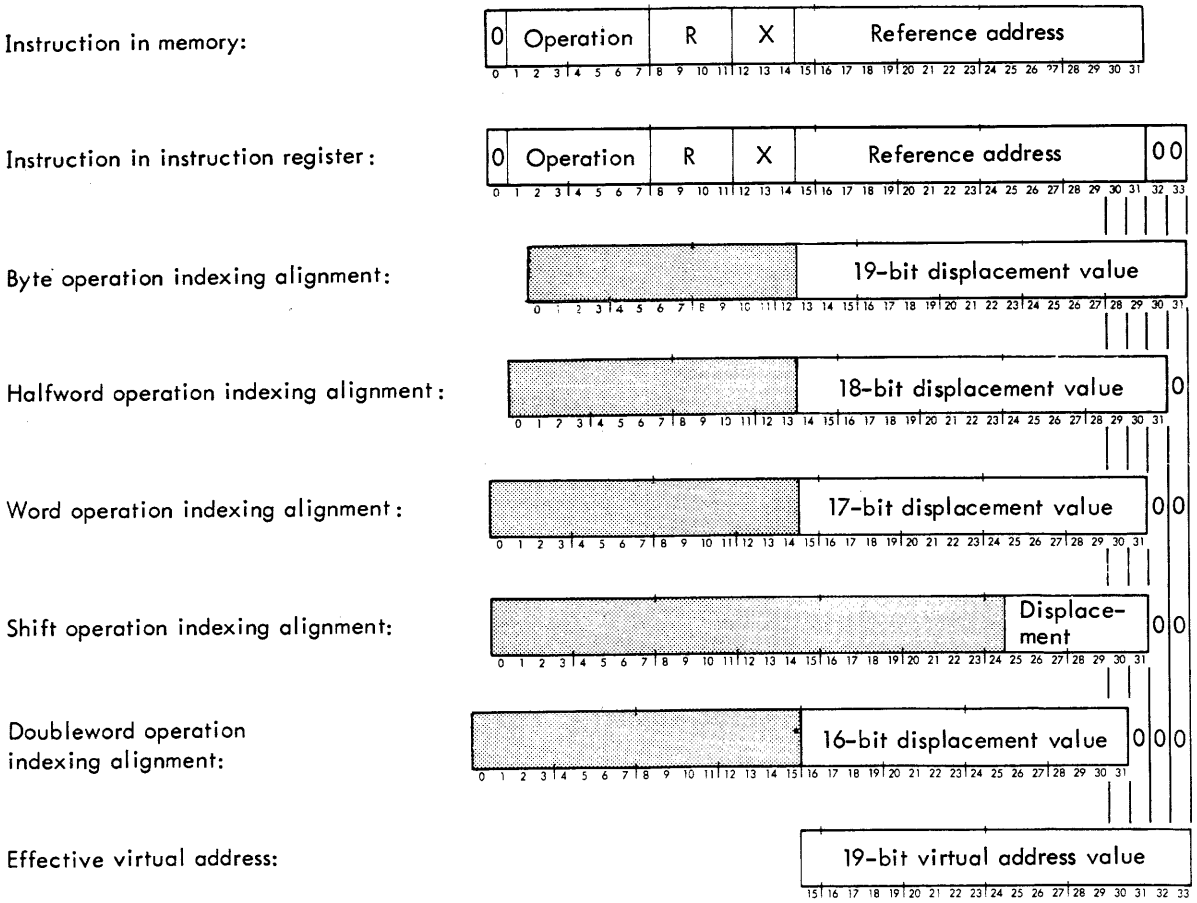


Figure 5. Index displacement alignment.

program is under control of the appropriate IOP with no further operations required by the CPU. CPU-I/O interaction is accomplished through I/O interrupts, the conditions for which are specified by the CPU in the I/O command list.

The IOP system operation is structured so that low cost, multiple channel, concurrent I/O operations which demand little CPU time for their execution are readily incorporated in the SIGMA 7 system.

Device Controllers and Devices. A wide range of 8-bit oriented peripheral equipment is available for attachment to IOPs. These include keyboard/printers high and low-speed paper tape input and output, punched card input and output, IBM compatible 7-channel multiple density magnetic tape units and single density 9-channel magnetic tape units, high speed line printers, fixed head rapid access disc storage units, communications equipment, and keyboard/display equipment. All such units are controlled by individualized Device Controllers which communicate with the IOPs through a common, simplified, electrical interface using a common method for control and information exchange.

Real-Time Interface Units. A full range of special systems equipment including such devices as Analog-to-Digital Converters, Digital-to-Analog Converters, and Multiplexers together with Device Controllers which interface them either with the direct input/output system of the CPU or with the standard IOP interface are also available.

SEVEN CRITICAL DESIGN PROBLEMS AND THEIR SOLUTION

General

This brief exposition of the SIGMA 7 system provides an over-all view of its principal features as a computing system, but it gives little insight into the special characteristics which uniquely permit it to carry out real-time tasks embedded in a multi-usage environment. Such an environment must be controlled by an executive program which allocates system resources; schedules operating intervals; provides services such as trap and interrupt response control, editing, compiling, assembling, and debugging; controls and executes I/O operations; swaps active programs between core and rapid access mass storage units; and guarantees the integrity, privacy, and non-interference of all active programs and their associated data bases.

If a real-time operation is to be maintained in a multi-usage environment, it must have guaranteed dedication and protection of the system resources which it requires. Core and disk space must be assigned to it and protected from access by other programs. I/O channels, peripheral devices, and interrupt levels must be assigned, dedicated, and protected from outside interference. The establishment of a real-time operation and the dedication of resources to it should be dynamically available through the operating system. These tasks must be accomplished in such a way as to permit full freedom and capability to the non-real-time operations while in no way degrading the responsiveness of the system to the time-sensitive demands of the real-time program. In the following section we will describe the design problems which were faced in meeting these requirements and present the SIGMA 7 structures which provide for their solution.

The Problem of Priority Interrupts

The system must be equipped with a true priority interrupt system which is flexibly structured and controlled and whose operation in establishing priorities and recording and sequencing interrupt requests is essentially instantaneous and independent of CPU action. Interrupts of higher priority must be permitted to interrupt partially completed responses to those of lower priority. To maintain fast response, interrupt requests should require no decoding action on the part of the CPU to determine their source or nature. Capability for dynamically varying the priority sequence to meet the demands of a changing environment must be available. *No other system element may be designed such that its proper operation requires the inhibition of the priority interrupt system for any period of time.*

The SIGMA 7 Priority Interrupt System. The SIGMA 7 interrupt system is best described from the ground up. The basic interrupt level has four mutually exclusive states which are designated as Disarmed, Armed, Waiting, and Active. A separate flip-flop is used to disable or enable the level (Fig. 6).

In the Disarmed state the interrupt level rejects all incoming interrupt trigger signals. In the Armed state the interrupt level will accept a trigger signal from an outside source, or from the CPU, and will

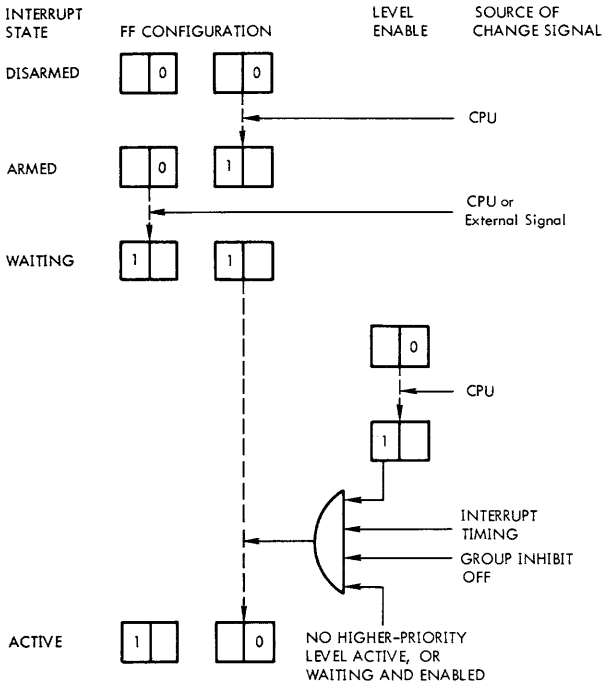


Figure 6. Interrupt level operations.

move to the Waiting state, where it will remain until the level is acknowledged by the CPU. If the level is disabled any Waiting condition is held in abeyance, preventing it from entering the priority chain of requests for CPU action. All Enabled and Waiting interrupt levels are permitted to enter the priority chain of requests awaiting computer interrupt response action.

Interrupt levels are organized into four classes which are designated as the Over-ride Class, the Counter Class, the I/O Class, and the External Class. The Over-ride class can never be Inhibited, Disarmed, or Disabled. A separate inhibit flip-flop is provided in the CPU for each of the other three classes, so that the CPU can prevent an entire class from entering the priority request queue. In effect this inhibit flip-flop disables the class regardless of the Enable-Disable states of the individual levels within it. The External Class is further divided into 14 groups each containing 16 interrupt levels. The priority request queue starts at the Over-ride Class and then may be threaded through the remaining Classes (and Groups of the External Class) in any order which the customer may desire. Thus, external interrupts may be given priority positions above, below, or in between those allocated to the Counter Class and the I/O Class (Fig. 7).

Each interrupt level has a unique location in low order memory dedicated to it. Control of the CPU is automatically forced to this location when the interrupt is acknowledged and permitted to move to the Active state. This action occurs whenever the highest priority Waiting, Enabled, and Uninhibited interrupt level is of higher priority than the highest priority currently Active interrupt level.

The CPU can control the states of the interrupt system. A group of sixteen interrupt levels are operated upon simultaneously under control of a sixteen bit mask which selects the subset of the sixteen to be modified. Operations which may be performed upon the mask-selected levels include Disarm, Arm and Enable, Arm and Disable, Enable, Disable, Load Enables, and Trigger. The Trigger function permits the CPU to apply an interrupt signal to its own interrupt system. This feature can be used to simulate an external interrupt environment for purpose of system checkout. It also permits the CPU to carry out the highly time sensitive portion of an interrupt response and then to create for itself a low priority interrupt to call for the deferred servicing of the less time sensitive portion at a less pressing time.

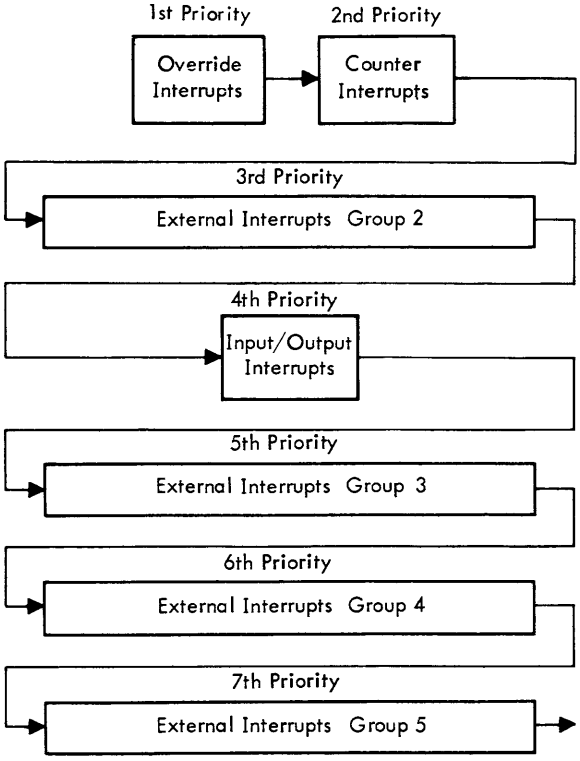


Figure 7. Typical interrupt priority chain.

The Problem of the Duration of Uninterruptible Intervals

Such an interrupt system is of little value if the CPU can remain for any significant period of time in an uninterruptible state. Under normal operating conditions, the longest uninterruptible interval must be kept short, and under abnormal conditions no malfunctioning peripheral hardware or software may be allowed to "hang up" the CPU in a noninterruptible state.

SIGMA 7 Interruptible Instructions and the Watchdog Timer. To insure that the longest uninterruptible interval which the CPU may experience in normal operation is short, all long instructions have been designed so that they may be interrupted *during* the course of their execution. Registers are held in fast memory, but instruction execution occurs in hardware elements. Since the original operands are retained in fast storage until instruction execution is completed, instruction aborting occurs without loss of information. Instructions whose duration is less than 10 microseconds are never aborted. Instructions in the 10-30 microsecond range are designed so that they may be aborted and subsequently restarted upon return from the interrupt. Instructions whose execution time exceeds 30 microseconds are designed so that they may be aborted and subsequently have their execution resumed from their point of interruption upon return from an interrupt process.

An instruction "watchdog timer," included in the standard SIGMA 7 configuration, guarantees against hardware hang-up by insuring that the time interval between interruptible points never exceeds 40 microseconds.

The Problem of Red Tape Time

Mere capability to initiate action in response to an interrupt is of little use to a real-time situation if it requires an inordinate amount of time to preserve the operating environment which exists at the time of the interrupt and to establish the new environment required for the processing of the interrupt. Hence, an extremely rapid context preservation and switching system must be provided in order to assure that minimum time lapse exists from the initiation of interrupt response to the execution of operations which are truly pertinent to the demands of the interrupt situation. Such a switching system must be repeatable to any number of levels in order to accommodate interrupts of interrupts.

SIGMA 7 Context Switching. A single instruction, Exchange Program Status Doubleword (XPSD) results in the collection of all of the active control states of the CPU and their storage in an arbitrarily designated doubleword location in core memory. This instruction execution then proceeds by loading the active control states with correspondingly structured information contained in the following two words in memory. Thus, the entire control environment of the CPU is stored and reloaded in six microseconds with the execution of a single instruction. A return to a prior control state is accomplished through the execution of another single instruction, Load Program Status Doubleword (LPSD), which also provides for clearing and arming or disarming the highest level active interrupt. An XPSD at the interrupt location saves the old environment and establishes the new one for the interrupt response. An LPSD at the conclusion of the interrupt process returns the CPU to its state prior to the interrupt. Since the storage and access locations designated by the XPSD and LPSD instructions are arbitrarily located in memory, nested chains of interrupted interrupt routines may occur to any level without loss of control and with automatic denesting as interrupt processes complete.

A second major element of context saving is the preservation of register states. Registers may be preserved in memory and restored through the use of multiple register load and store instructions or may be preserved in core implemented stacks through the use of multiple register push and pull instructions. Even the high speeds of these operations may result in too great an overhead time for some real-time processes; hence, a register storing and loading technique which is accomplished during the execution time of an XPSD instruction is provided. This technique is available whenever the CPU is equipped with one or more of the optional additional register blocks. The 5-bit Register Pointer is a portion of the contents of the Program Status Doubleword which is stored and loaded with the XPSD instruction. Hence, if a register block is available and dedicated to a real-time process the execution of the XPSD instruction which initiates the process automatically preserves the control context and the register context of the interrupted routine and automatically establishes the corresponding contexts for the interrupt process. Under these circumstances, the equivalents of register preservation, loading, and restoring are all accomplished within the execution

times of the XPSD and LPSD instructions which initiate and terminate the interrupt routine (Fig. 8).

The Problem of System Integrity

Some means must be provided to guarantee the integrity of the executive system and for it, in turn, to establish and guarantee the integrity of all other programs.

Master/Slave States and Privileged Operations. The SIGMA 7 CPU can operate in either a Master or Slave state. In the Master state all instructions can be executed normally. In the Slave state instructions whose execution are critical to the integrity of system resources are illegal. Such instructions are designated as Privileged Operations and are reserved to programs operating in the Master state. Privileged operations include all instructions which affect Input/Output operations through the Input/Output system, Input/Output operations direct to memory, the memory protection systems, the interrupt system, the operating state of the CPU (e.g. a Slave state program cannot switch itself to the Master state), or the continuation of system operation.

Memory protection, the other aspect of guaranteeing system integrity, is presented in the following section.

The Problem of Space Sharing

Efficiency in multiusage implies the simultaneous residency of many programs, or portions of programs, so that when conditions require that control be given to a new program it is resident and such action can occur immediately. Thus, under control of the executive system, partially executed programs must be permitted either to space share or to be swapped out of memory and later returned, preferably to whatever space is available. When a program is held up for I/O operations, only its I/O buffer region should be retained in core with the remainder of the program dumped to disk so that its space in core may be available for other usage. As such actions take place, the available memory space rapidly becomes fragmented into discontiguous regions which should be directly usable without having to repack the memory in order to achieve contiguity. Thus, a system should be provided for the execution of programs which have been dynamically relocated into discontiguous memory regions.

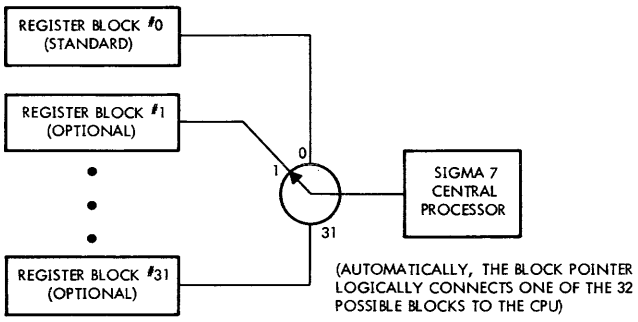


Figure 8. Block pointer and register selection shown with a block pointer value of 1 (00001).

The SIGMA 7 Memory Map. Dynamic program relocation into discontiguous fragments of memory is provided through the incorporation of an optional feature, the memory map. If the map option is installed, any program may be broken into 512-word pages and distributed throughout the implemented core memory in whatever 512-word pages of space are available. The memory map then permits the program to be executed as though it were located in the contiguous region of memory for which its addresses have been established. Clearly, the map provides the transformation of Virtual Addresses (i.e., addresses generated within a program such as instruction addresses, operand addresses, and indirect addresses) into Real Addresses (i.e., the physical core addresses where program-designated values are actually located).

The memory map employs a 256-byte, integrated circuit memory in its implementation, and thus provides for the mapping of a full 128K Virtual Address space. A mode flip-flop designates whether a program is to operate in a mapping or non-mapping mode. When mapping is invoked, the following events occur every time an actual reference to memory is to be made (Fig. 9):

1. The 17-bit address generated by the program is broken down into a 9-bit word address and an 8-bit page address.
2. The 8-bit page address is used to access one of the 256-byte map memory locations.
3. The 8-bit page address stored at that location replaces the 8-bit page address portion of the Virtual Address to form a Real Address.
4. The Real Address is used to access the memory.

Because of the speed of the integrated circuit memory, these actions add only 60 nanoseconds to each memory access.

A special instruction, Move to Memory Control, provides for rapid changing of the memory map.

With the map option, a program can be brought in and distributed to any set of 512-word pages which may be available in memory. The Move to Memory Control instruction is then used to write the map so that the proper address transformation will be made. The mapping mode is then entered and control is turned over to the program, which then operates as though it were located in the contiguous region of memory for which it was designed. The operation of such a program may be halted at any time, the program subsequently relocated to any other set of 512-word pages, the map rewritten, and the program operation resumed with no adverse effects.

Programs whose addresses range over the 128K Virtual Address space may be executed on a machine with far less than 128K words of implemented core. The map permits portions of such programs to be resident and operate in available core space. Pro-

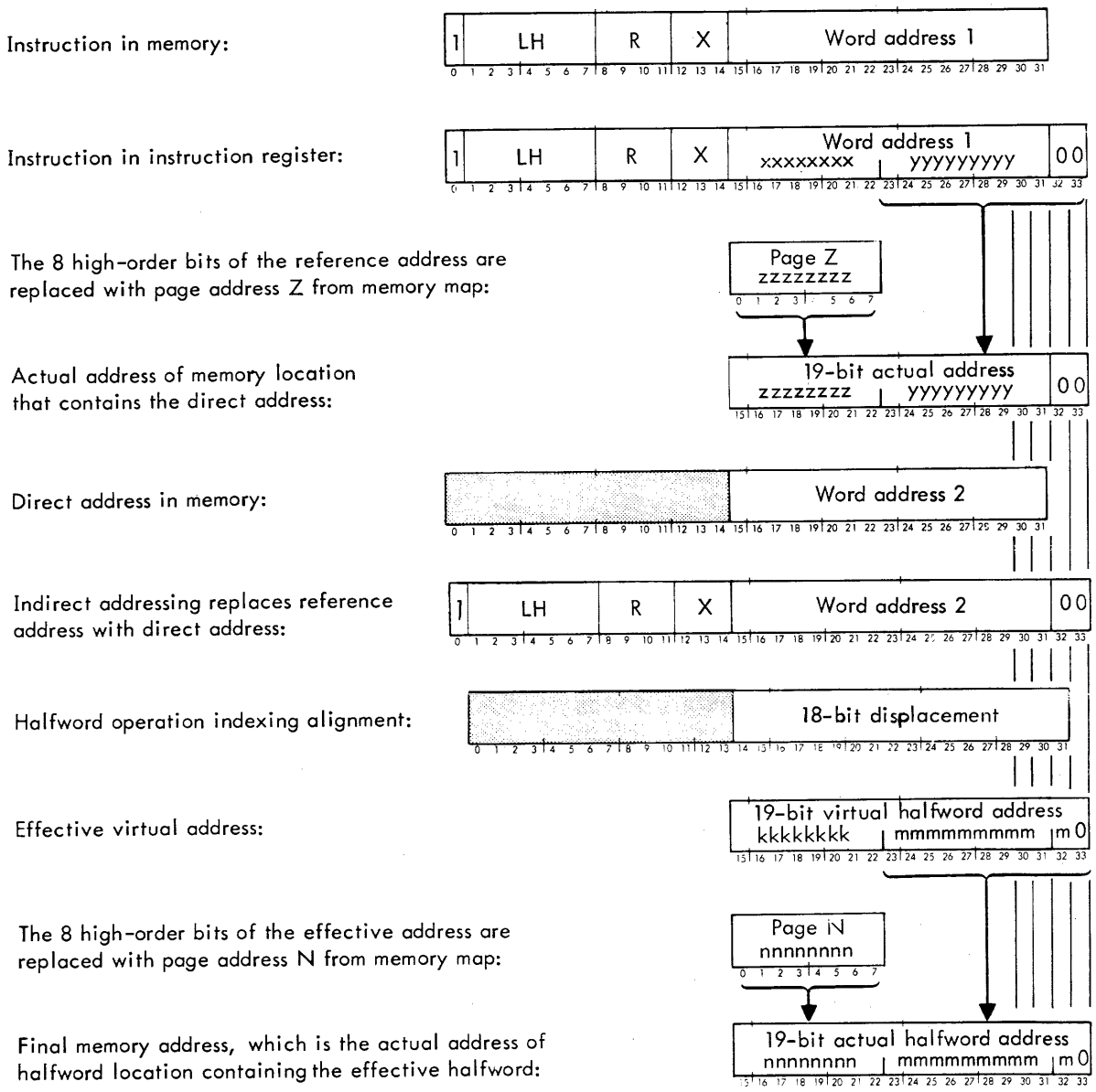


Figure 9. Example of generation of actual memory addresses; indirectly addressed, halfword operation.

gram references to blocks which are not resident are automatically trapped so that a page-turning system may be readily implemented.

A number of design compromises were made in the incorporation of the map in SIGMA. The most important of these was the decision not to incorporate a two-level (segment and page) mapping structure. Consequently, all programs which must directly communicate with each other, without the intervention of the executive system, must share a common Virtual Address space since they must share a common map. This includes the executive system itself which must provide services to user programs. When doing so, the executive system must operate in the mapping mode since no unique bit was available in each instruction word to designate whether or not to employ mapping on an individual instruction execution basis. Thus, in the interests of simplicity and limitation of costs, the map system has been deliberately incorporated in such a way that a user's Virtual Address space is curtailed by the size of the executive system and the public routines and services to which the user's program desires to have access. Further, these latter programs must have dedicated space in the Virtual Address space of all users who desire to use them so that they may maintain constant residency in all users' maps. While these limitations were recognized it was felt that it was worth far more to achieve the powers of the mapping operation at a price which would bring them to a large segment of the market, than it was to achieve full segmentation for a much smaller portion of the market.

The Problem of Memory Protection

An additional aspect of guaranteeing the integrity, privacy and non-interference of all active programs is that of memory protection. Early implementations of memory protection were aimed almost exclusively at providing the write protection function which is essential for guaranteeing that one program cannot destroy another. The multi-usage environment adds further dimensions to memory protection requirements. Privacy considerations of privileged information (such as payroll data) require that portions of memory be protected from unauthorized reading as well as writing. The complexity of the operating environment makes it highly desirable to catch errant programs at the earliest possible time. This desire leads to the concept of instruction protection which prevents a program from executing an instruction taken from an instruction-protected region of memory.

Access Protection. An additional 512 bits of fast memory are supplied with the map option. These provide storage for two Access Protection bits which are associated with each of the 256 Virtual Address pages. These bits are accessed during the mapping operation whenever the CPU is in the Slave state. They are used to impose inhibitions on a slave program's use of the information in the page which it is attempting to access. These inhibitions are designated in the following table:

Access Protection Value	Inhibition/Permission Control
00	Permit slave access to this page for any purpose
01	Inhibit slave access for writing, but permit instruction or operand read access
10	Inhibit slave access for writing and instruction execution, but permit operand read access
11	Inhibit slave access for any purpose

Note that these inhibitions are imposed on slave Virtual Addresses and are in effect no matter where the slave program may be located physically in core.

The Access Protection bits are used to restrict the operation of a slave program to its allocated addressing domain, and within that domain to permit the establishment of read-only or read-and-execute only pages of information. Thus, provision is made to guarantee secrecy and preservation of sensitive information, for common use of non-destructible data bases and public subroutines, and for the trapping of run-away program attempts to execute data.

The 64-byte Access Protection fast storage area is loaded with a Move to Memory Control instruction. Because of the existence of this form of program inhibition, the memory map need only be loaded for the address domain over which a slave program is expected to operate. The Access Protection bits are loaded for the full 128K Virtual Address domain and thus are guaranteed to protect against slave program operations in pages outside their prescribed domain. This fact reduces overhead time involved in map loading for slave programs with restricted addressing ranges.

Memory Write Protection. The Access Protection bits operate over the Virtual Address domain, are effective only for slave programs, and are not available unless the Memory Map option is installed. Consequently, an optional memory write protection feature which operates independent of the Access Protection bits is also available. The memory write protection feature operates in both the Master and Slave states. This feature is implemented with a 512-bit fast memory unit which stores a 2-bit write protection "lock" for each 512-word page. Every operating program is given a 2-bit "key" which, in conjunction with the locks, controls its write access to a page in the memory according to the following rules:

1. If the lock value for the page is 00, writing is unconditionally permitted. That is, the page is "unlocked."
2. If the key value for the program is 00, writing is unconditionally permitted. That is, the program has been given a "skeleton key."
3. If the lock and key values are both non-zero, then writing is permitted if, and only if, the lock and key values are identical.

Note that this feature is associated with the use of Real Addresses and, therefore, supplies write protection for physical memory. If the map option is installed, both forms of memory protection are operative, the Access Protection bits operating on the Virtual Address space of the program and the locks and keys on the physical memory space, after mapping. Locks are installed through the use of the Move to Memory Control instruction. The memory write protection system makes it possible to provide memory protection in the absence of the Memory Map. It also provides memory protection for simultaneously resident Master mode programs, thereby guaranteeing their integrity and the integrity of publicly available, reentrant, pure procedures which service users of both classes. This form of memory protection also provides a powerful tool for the development or revision of portions of the executive system. Such a development can occur on-line, while the system is operating, since the unchecked portion can operate under a write protection constraint which guarantees the memory integrity of the system.

The Problem of Recursive and Reentrant Routines

Efficient operation in a multi-usage environment requires efficient utilization of memory and minimization of program swapping time. The provision of single, public copies of routines which are used in common by many concurrently operating programs is an essential ingredient in optimizing both of these functions. Public routines avoid multiple copies, one for each user, and eliminate the swapping time associated with their transmission between core and rapid access disk. (Indeed, swapping out time is always avoided for all pure procedures.) Public routines must be pure procedures which operate on a designated context. When the context and working space are provided by the calling program and several such programs may be concurrently using the routine it is said to be reentrant. When such a routine may repeatedly call itself, and, therefore, be required to provide its own nested context and working space, it is said to be recursive. A single routine may be both recursive, i.e., capable of calling itself, and reentrant, i.e., capable of being called by many different programs prior to its completion of operations for any single one of them. Of primary importance is the requirement that public routines must operate in an interrupting environment in which they may be invoked by one or more programs before completing their operations for another. The primary SIGMA 7 design constraint that no software may be designed so that it must turn off the interrupt system in order to be guaranteed to operate properly is particularly difficult to meet in this environment. These requirements place demands on the hardware to provide entry and context establishing methods which provide for efficient and dynamic utilization of space and guard against loss of information or control under all operating conditions.

SIGMA 7 Hardware Features for Reentrance and Recursion. Both reentrance and recursion require an efficient means for guaranteed preservation of the context of a partially completed process, including the 16 general registers and the link address, and for the institution of the corresponding context for a new user. A Branch and Link instruction which preserves the program address in a designated register provides a simple and effective linking mechanism for both reentrant and recursive routines. The indirect addressing and indexing mechanisms provide one of the means for context designation in the reentrant case. Multiple register blocks provide a rapid means

for context switching. The memory map provides the most direct and effective method for both context designation and context switching since it permits the reentrant routine to directly address its designated context. Switching from context to context then merely requires a map change.

Both types of programs require register preservation. The Load Multiple and Store Multiple instructions provide a ready solution for the reentrant case since the calling programs must provide storage space within their context regions. The recursive case is more complex since the recursive program must provide its own storage. In this case, however, such storage is always used in a nested fashion on a last-in-first-out basis. The SIGMA 7 pushdown stack manipulating instructions provide the ideal solution for this situation. There are five instructions in this set, PUSH, PULL, PUSH MULTIPLE, PULL MULTIPLE, and MODIFY STACK POINTER. These stack manipulating instructions provide an efficient means for moving information between single or multiple registers and core locations which are contained within a pushdown stack which is under control of a doubleword stack pointer (Fig. 10).

The stack pointer contains the address of the top of the stack, a count of the words in the stack, a count of the number of spaces currently available in the stack, and stack underflow and overflow trap mask bits. With such a mechanism, recursive entry to a routine merely requires the execution of a single PUSH MULTIPLE instruction to preserve the current context in a stack for which space is provided within the routine itself. A routine which is both reentrant and recursive merely uses pushdown stacks which are stored within the context regions of the various calling programs.

In general, the pushdown stack mechanism provides a powerful tool for any dynamic space allocation situation in which a last-in-first-out nesting of information is guaranteed.

Other Multiusage Features. The limitations of space do not permit the description of many of the details of the SIGMA 7 which make for efficiency of operation in a multiusage environment. A few additional features are worthy of mention, however. A

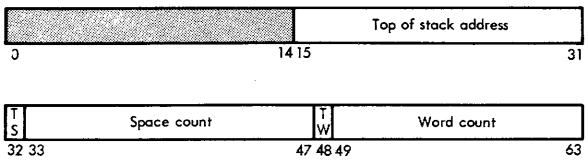


Figure 10. Stack pointer.

set of four Call instructions, each providing sixteen independent branches, generate a total of 64 generalized operator or subsystem entrances. The Call, operating through the Sigma Trap system and the use of XPSD instructions, provides a mechanism for accessing generalized, re-entrant service routines. The Call mechanism provides the proper control states for these routines and establishes the means for returning to the control state of the calling routine, without going through the executive program and without using the address portion of the Call instruction itself. Consequently, the address portion of the Call instruction is available for the designation of operand(s) to the called routine. Calls thus can be considered a generalization of the SDS Programmed Operator concept.

The SIGMA 7 CPU is equipped with two real-time counters, and 60 cycle, 1KC, 2KC, 4KC, and 8KC clock sources, any of which may serve as inputs to the counters. Optionally, it may be equipped with two additional counters. Clock pulse counting is handled by single instruction interrupts which cause counts to be accumulated in arbitrarily designated memory cells. Overflows from these locations cause a second interrupt, unique to each counter, to occur. Hence, real-time synchronization may be maintained, elapsed time intervals may be measured, and arbitrary length count down timers may be established, all without recourse to elaborate, software-derived timing routines.

An extensive error-trapping system provides for automatic error detection and recovery from situations which would otherwise eliminate all possibility of interrupt responsiveness.

An optional power fail-safe system provides for the detection of incipient power failure and the orderly shut-down of the system so as to preserve its operating state. An automatic start-up procedure is initiated upon restoration of power.