Sperry Univac

Computer Systems

G. Bell, S. H. Fuller, and D. Siewiorek, Editors

The Evolution of the Sperry Univac 1100 Series: A History, Analysis, and Projection

B. R. Borgerson, M. L. Hanson, and P. A. Hartley
Sperry Univac

The 1100 series systems are Sperry Univac's large-scale mainframe computer systems. Beginning with the 1107 in 1962, the 1100 series has progressed through a succession of eight compatible computer models to the latest system, the 1100/80, introduced in 1977. The 1100 series hardware architecture is based on a 36-bit word, ones complement structure which obtains one operand from storage and one from a high-speed register, or two operands from high-speed registers. The 1100 Operating System is designed to support a symmetrical multiprocessor configuration simultaneously providing multiprogrammed batch, timesharing, and transaction environments.

Key Words and Phrases: 1100 computer series, computer architecture, multiprocessing, multiprogramming, operating system, programming languages, data management systems, end user facilities, executive control software

CR Categories: 1.3, 4.0, 4.20, 4.30, 4.32, 4.33, 4.35, 6.0, 6.21, 6.30

Authors' Addresses: B. R. Borgerson, Sperry Univac, P.O. Box 500 M9-114, Blue Bell, PA 19424; M. L. Hanson, Sperry Univac, P.O. Box 3942 MS 4243, St. Paul, MN 55165; P. A. Hartley, Sperry Univac, P.O. Box 3942 MS 4823, St. Paul, MN 55165.

1. Introduction

Although there are earlier Sperry Univac® systems with the 1100 designation, the 1107 system represents the first of the compatible 1100 series. The 1107 system, first delivered in 1962, marked the introduction of the basic structure of the current 1100 series architecture. The only major features carried forward from the pre-1107 systems were the 36-bit word and the ones complement arithmetic. Although the basic architecture has remained the same since the 1107, the series has progressed through 4 architectural classes, 8 unit-processor systems, and 23 different processor/system configurations. Over 1200 of the 1100 series processors had been delivered as of July 1977. These systems constitute a worldwide installed base which at approximately 4 billion dollars, is second in value only to the IBM 360/370. The evolution of the 1100 series is depicted in Figure 1 which shows the date of introduction of each system. Figure 2 shows the genealogy and depicts the four architectural classes.

Most user programs written on any post-1107 series system will run on any other 1100 series system without modification. The primary exceptions are time-dependent programs and certain user generated, foreign device specific, assembly language code.

The 1107 was originally designed for batch-oriented scientific and engineering applications. The architectural and the operating system evolutions reflect a continuing push towards more efficient interactive and business-oriented capabilities. The change in the relative percentages of scientific work versus business-oriented work of current systems reflects this trend. Substantially more than half the total current work loads on 1100 systems are business oriented.

One of the strongest features of the 1100 series is its multiprocessor capability. Multiprocessor configurations have been in general use since the introduction of the first 1108 multiprocessor system in 1968. We believe the 1108 was the first commercially available general-purpose computer to support a completely symmetrical multiprocessor system; i.e., all processors coequal in sharing the same memory, i/o channels, and a single copy of the Executive System. Early 1100 multiprocessor systems did not yield the expected performance improvement over unit processor configurations, and hardware and software stability problems were encountered in the first systems. The feedback of many years of experience has produced solutions to these early problems. As a result, high efficiency, stability, and system availability are currently being achieved with symmetrical multiprocessing configurations. As indicated in Figure 1, the multiprocessor configurations of the 8 base systems increase the total number of systems in the 1100 series to 23.

To limit the length of this paper and to focus more sharply on the 1100 series architecture and operating system, three important areas are omitted: the com-

Fig. 1. 1100 Series evolution.

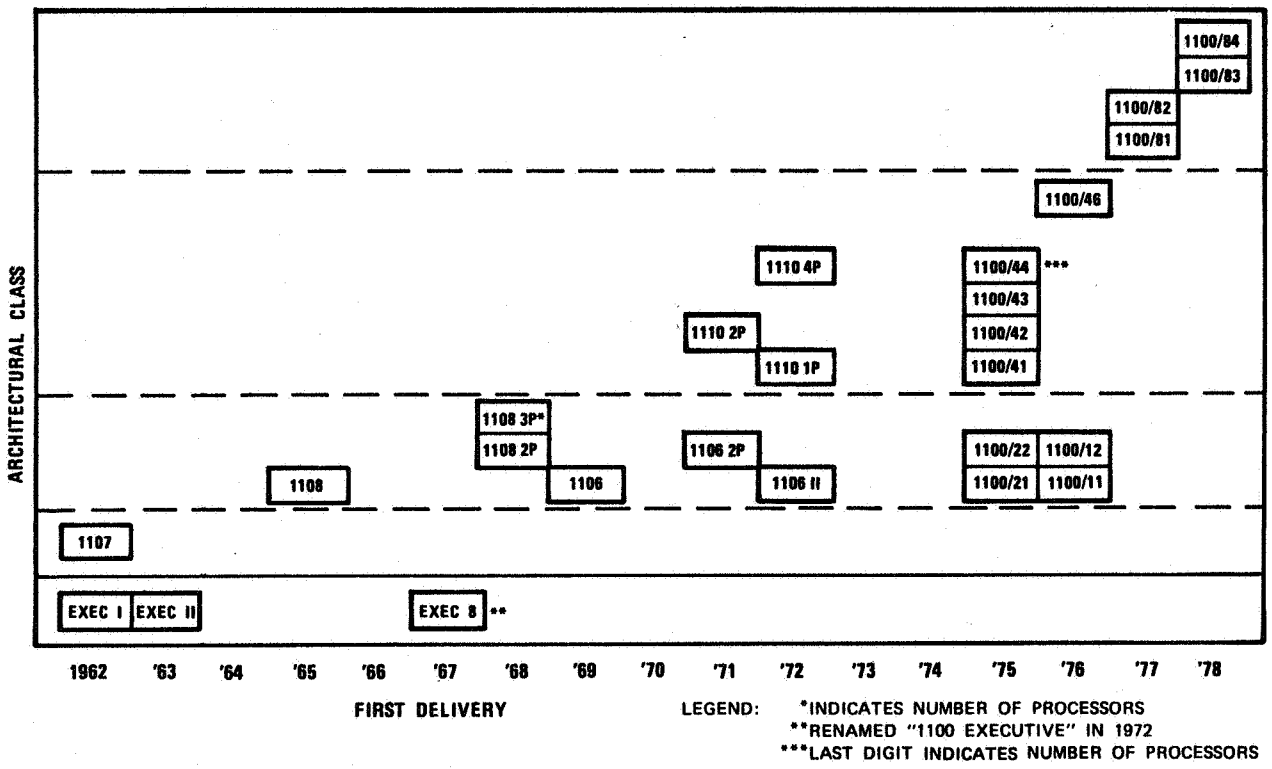
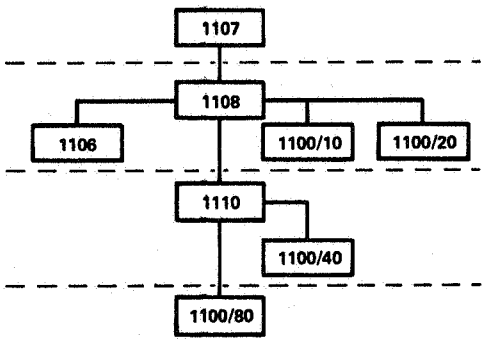


Fig. 2. 1100 Series architectural genealogy.



munications front-ends; the peripherals associated with each system; and the available applications programs. As a result, this paper is divided into two sections. The first section provides a brief summary of the basic 1100 series structure through a description of the 1107 architecture, discusses the architecture and technology of each of the 1100 series systems, and examines several possibilities for future 1100 series systems. The second section follows a similar progression for the 1100 series operating system. For ease of comparison and contrast, a summary of some pertinent characteristics of the 1100 series systems is given in Table I.

Expressing system speed in units such as millions of instructions/second is of dubious value unless the reader is familiar with the instruction repertoire of the system and the mix of instructions used. Consequently, this paper uses the basic add time as a metric. In the basic add, all systems in the 1100 series read a 36-bit operand from storage (indexed if desired), add the value to an accumulator register, algebraically increment the index register if specified, read the next instruction, and are ready to execute this next instruction when the add has been completed. In addition to the add time, Table I lists the time to do a single-precision, floating-point divide which provides a better indication of the relative processor speeds since the divide time is relatively independent of memory cycle time. The slowest user instruction tends to be a double-precision, floating-point divide but the use of this instruction would exclude the 1107 system from the comparison.

2.0 Architecture and Technology

2.1 1107 System

The 1107 system, introduced in 1962, provided the base architecture for the 1100 series. The 1108, 1110, and 1100/80 descriptions begin with a summary of major changes from the previous system.

The major features of the 1107 are:

- 36-bit word, ones complement arithmetic
- 16 index registers, 16 arithmetic registers
- standard usage is one memory operand and one register operand; optional usage is two register operands
- unlimited cascading of indirect addressing, with indexing possible at each level
- 16-bit address field plus index yields absolute address; 64K words maximum for memory
- interrupts are to a dispatch table in memory
- two memory write lockout windows (however, no privileged state was provided)

Table I. 1100 Systems characteristics.

	1107	1108	1106	1110	1100/20	1100/40	1100/10	1100/80
First Delivery	1962	1965	1969	1972	1975	1975	1976	1977
Integer Add Time (ns)	4000	750	1000[1]	300[2]	875	300[2]	1125	200[3]
F.P. Divide Time (ns)	26700	8250	11000[4]	5200[2]	8325	5200[2]	8625	4800
Number Base Registers	0	2	2	4	2	4	2	4
Maximum Number Processors	1	3	2	4	2	6	2	4
I/O Channels/ Processor	16	16	16	0	16	0	16	0
I/O Channels/ IOU	–	16	–	24	–	24	–	26
Maximum Number IOUs	0	2	0	4	0	4	0	4
Storage Structure	1 level	1 level	1 level	Primary/ Extended	1 level	Primary/ Extended	1 level	Cache/ Main
Logical Address Space (words)	65K	262K	262K[5]	2^{30}	524K	2^{30}	524K	2^{30}
Physical Memory Capacity (words)	65K	262K	262K[5]	262K/ 1M	524K	524K/ 1M	524K	16K/ 4M
Memory Technology	Core	Core	Core	Wire/ Core	MOS	TTL/MOS	MOS	ECL/MOS
Number Processors By 8/1/77	36	296	338	290	50	105	100	12
Number Instructions	117	151	151	206	151	206	151	201[6]

Notes: [1] 1500 ns with 262K of "Unitized" Storage. [2] Time for executing out of primary storage. [3] Time assumes in "hit" in cache. [4] 11,500 ns with 262K of "Unitized" Storage. [5] At initial delivery; Can be field-upgraded to 524K. [6] Does not include 494 system instructions.

- input/output performed via 16 bidirectional word channels
- single-precision, floating-point computational capability provided
- trace mode provided for aiding program debugging
- two subroutine call instructions provided: one call instruction leaves the return address in an index register, the other leaves the return address at the memory location ahead of the subroutine
- an extensive test and search repertoire is provided, including instructions to test, search, and mask search for an operand within or outside specifiable limits

Fundamental to the 1100 architecture is a General Register Set (GRS) that is structured as shown in Figure 3. Most instructions perform some operation on, or examine, a value located in one or more (up to four) of these high-speed registers. In addition to the user visible registers, the GRS contains information such as input/output controls; however, these additional entries are not of direct interest to a user.

2.1.1 Instruction Format

The instruction format is shown in Figure 4. The two 4-bit fields, a and x, allow specification of two registers out of the GRS. The GRS is composed of three subsets of registers: X registers are used for indexing; A registers are general-purpose arithmetic registers; R registers are intended for special purposes to be discussed later.

The X registers have the format shown in Figure 5. If indexing is not specified (the instruction x-field contains all zeros), the operand address in main storage is the 16-bit u-field. If the x-field is nonzero, it selects 1 of the 15 X registers and the effective operand address is $u + X_m$. Additionally, if the h-bit is 1, the operation $X_i + X_m \rightarrow X_m$ is performed, effecting a programmer-defined increment or decrement (X_i is

Fig. 3. General register stack.

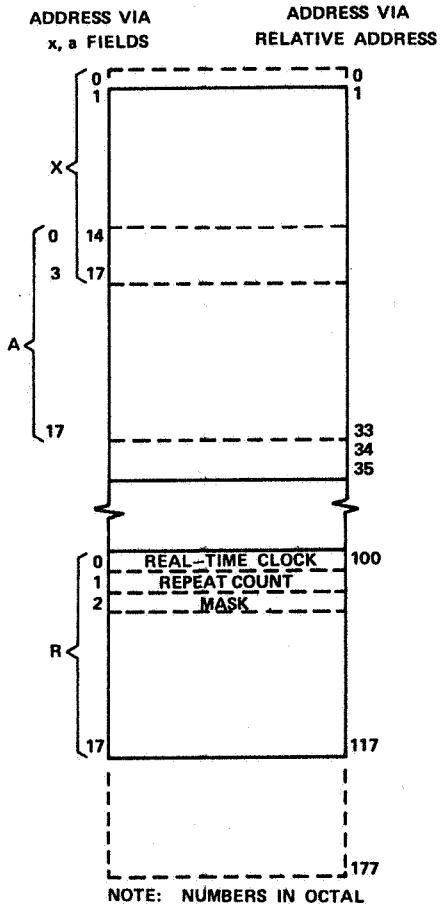


Fig. 4. Instruction format.

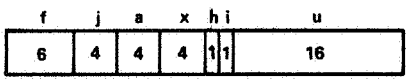


Fig. 5. Index register format.

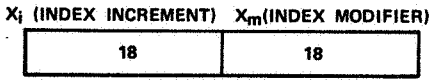
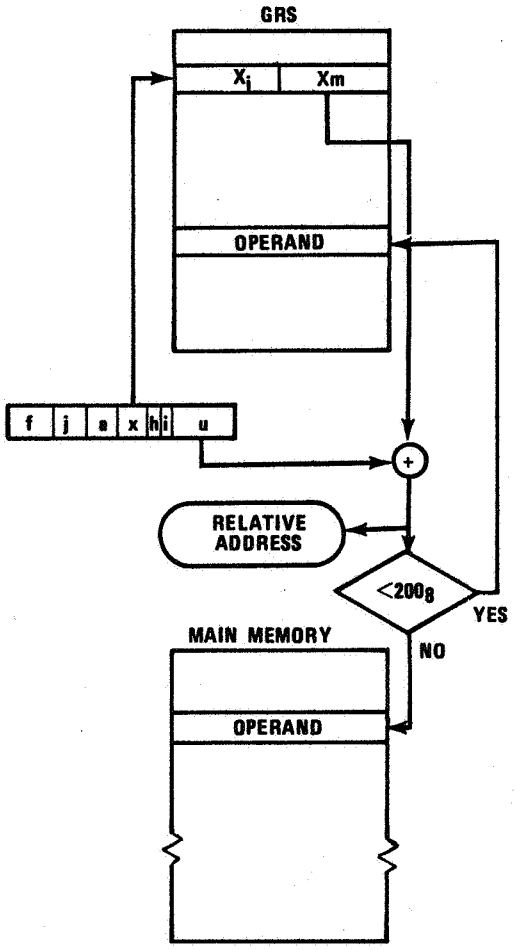


Fig. 6. 1107 Addressing.



treated as a signed value) of the index. This permits, for example, the summation of an array in any single dimension with a 2-instruction loop, where most processors would require a 3- or 4-instruction loop. There is no overhead for either indexing or incrementing the index. Register to register transfers are accomplished by specifying one register in the a-field and the other in the u-field (relative address less than 200_8). By operating system convention, all user programs are biased so that they never have a main storage address less than 1000_8; consequently, users are not impacted by seemingly ambiguous addresses in the range 0–200_8. The 1107 addressing is shown in Figure 6.

The arithmetic register to be used is specified by the 4-bit a-field. This is usually one of the A registers but it can also be an X or R register; there are some special instructions for each of the register subsets. Note in Figure 3 that the last four X registers overlap the first four A registers, thus making a more powerful instruction set available for some of the index registers. Parameters are usually exchanged with procedures by leaving values or addresses in the first few A registers.

The i-bit (see Figure 4) provides cascaded indirect addressing. When the i-bit is zero, an operand is fetched as described above. When the i-bit is 1, an indirect word is fetched which replaces the original x, h, i, and u fields. If the i-bit of the word fetched is also 1, another word is fetched. Since each fetch has independent control of x, h, i, and u, chains of considerable complexity can be formed. This cascading process continues until eventually a word is fetched with $i = 0$; this word is used to compute the operand address.

The f-field (see Figure 4) defines the instruction to be performed. For some values of f, the j-field is used as a minor function code. The j-field is also used to select a 6-, 12- or 18-bit byte out of memory and right justify the byte in an arithmetic register (the 1108 expanded this usage to accommodate 9-bit bytes). A third use of the j-field is to specify an immediate operand. If x is zero, the immediate operand is h, i, u; if x is nonzero, the immediate operand is $u + X_m$. The immediate operand can be specified either with or without sign extension; i.e., bit 17 is propagated throughout bits 35 through 18 (higher order bits).

2.1.2 Arithmetic Instructions

Figure 7(a) shows the format of a single-precision, floating-point word. The exponent value ranges from -200_8 to $+177_8$. A bias of $+200_8$ is added to all exponents to form a characteristic in the range from 0 to $+377_8$. This format allows the use of fixed-point instructions for making magnitude comparisons on floating numbers. Instructions to aid conversion between integer and floating-point forms are provided, as well as four single-precision, floating-point instructions. The mantissa contains 27 bits.

Negative numbers are formed by performing a ones complement transformation on the representation of the positive value of the number. Normalization is binary so that all bits are meaningful, providing approximately 8 decimal digits of precision. The range of floating-point numbers is approximately from 10^{38} to 10^{-38}.

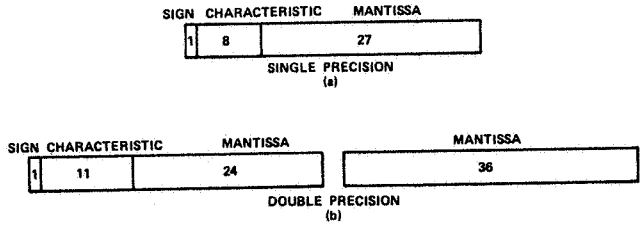
The ones complement arithmetic has a useful side effect in that the arithmetic complement instruction also achieves logical negation. Thus *A operator B* $\rightarrow$ *C*, where *operator* is any of the 16 possible logic functions, performs the operation in a maximum of three instructions.

2.1.3 Iterated Instructions

There are iteration instructions that use the *R* registers; in particular R_1 is used as a repeat count register and R_2 is used as a mask register. One of these instructions can use any *R*-register (except R_0 which is used for the real time clock). The use of multiple *R*-registers allows efficient control of nested loops.

Many search and masked-search instructions are provided. A particularly powerful instruction, masked search for operand within limits, will be used for illustration purposes. This instruction causes a search to determine if an operand is within a predefined range. The range is specified by preloading the lower and upper limits into two consecutive *A* registers. The *a*-field in the search instruction points to the first of these registers. For masked, iteration instructions, R_1 contains the maximum number of operands to be tested and R_2 contains the mask that specifies which bits are to be compared. Since $X_m + X_i \rightarrow X_m$ on each iteration, flexibility is provided for efficiently searching multidimensional arrays. If a compare is successful then the repeat count is decremented, the next instruction is skipped, and X_m can be used to identify the operand that compared successfully. If a compare is unsuccessful then the repeat count is decremented, X_m is modified, and the next compare is made. If the repeat count reaches zero, the next instruction is executed. Note that R_1 and R_2 are implicitly defined by the function code as opposed to a register selection field within the instruction word. In addition to the search instructions, a block transfer instruction uses a similar process to provide fast storage to storage transfer.

Fig. 7. Floating-point formats.



2.1.4 Other Instructions

Two jump instructions are provided to allow subroutine calls. One of these instructions stores the return address in an index register thus allowing pure-procedure code. The other instruction jumps to the entry point of the procedure and stores the return address at this address minus one. This instruction prohibits the use of pure-procedure but allows more convenient nesting of subroutines.

An instruction is provided for implementing a primitive memory lockout mechanism. Two writable windows can be specified. These windows both start and end on 4K-word boundaries. Since no privileged-mode exists in the 1107, this instruction is only useful in protecting a user from himself. For debugging purposes, a trace-mode instruction is provided which puts the processor in trace mode and causes an interrupt every time a branch is about to occur (either unconditional or successful conditional).

2.1.5 Program Organization

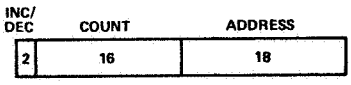
Physical storage is organized into two independent modules allowing two independent accesses concurrently. To make use of this capability, programs are assembled using two logical "location counter" groups. One of these location counter groups is usually devoted to instructions and the other to data, to permit fetching instruction $n + 1$ simultaneously with the fetching of the operand for instruction n. The storage units have a cycle time of 4 microseconds but the overlap allows an effective cycle time of 2 microseconds.

2.1.6 Input/Output

There are three basic instructions provided to initiate an input/output sequence. Once initiated, the channel hardware completes the transfer without intervention by the processor. The *a*-field of the instruction word specifies 1 of 16 input/output channels, $u + X_m$ points to an input/output control word formatted as shown in Figure 8, and the *f*-field specifies the instruction: Load Function in Channel (send command words to the peripheral); Load Input Channel (read); or Load Output Channel (write).

The channel uses an input/output control word to control the transfer. If a channel is in output mode (a Load Output Channel instruction was executed) and a request for a word is received from a peripheral, the channel will transfer the word in main storage specified by Address to the peripheral, increment or decrement

Fig. 8. Input output control word.



the Address field (as defined by the Inc/Dec field), and decrement the Count field. If a channel is in input mode and a signal is received indicating that the peripheral has a word available, the channel stores the word at the Address specified in the input/output control word, increments or decrements the Address field, and decrements the Count field. Variants of the above three instructions cause an interrupt to occur when the Count goes to zero. Data transfers take the form of 36-bit words transmitted over a dedicated channel for each peripheral subsystem. The 1107 system can have a maximum of 16 of the these bidirectional channels which are an integral part of the processor.

Channel priority is based primarily on the function being performed on the channel. In some cases, the priority is dynamic so as to avoid a total lockout of some functions. For example, input data requests have priority over output data requests; however, if an input transfer is in progress and both input and output requests are pending, the output request will be given priority.

2.1.7 Technology

The 1107 processor is constructed with discrete semiconductor technology except for the GRS, which uses a thin film storage element. Main storage is core memory with a 4-microsecond cycle time giving a basic add time of 4 microseconds.

2.2 1108 System

The 1108 system, first delivered in 1965, introduced several improvements over the 1107. The major improvements were increased throughput and enhanced protection in multiprogramming environments. The primary new feature which provided these improvements was a relative addressing structure which created a dynamic relocation capability. In addition to faster unit processor operation, multiprocessor configurations were introduced which offered higher performance and greater system availability. Significant new features on the 1108 are:

- changes to support multiprocessor operation
  - test and set instruction included for interprocess communication
  - interprocessor interrupt capability
- relative addressing achieved via two base registers
- better storage protection as part of relative addressing structure
- logical address range increased to 65K words for programs, and 262K words for data
- privileged mode
- double-precision, fixed-point instructions
- double-precision, floating-point capability
- *a*-field used as minor function designator to extend the number of function codes
- faster instruction execution
- availability control unit to partition components into three systems
- memory switches for multiple processor access to each memory module
- shared peripheral interface to provide switching between multiple processors and peripherals
- 1107 compatibility mode to invoke 1107 addressing
- physical storage increased to 262K words

2.2.1 Addressing

A major improvement to the 1107 architecture occurred in the area of addressing. The 1108 hardware allows the instructions and data of any program to be dynamically relocated in any available space. This is made possible by the introduction of two base registers. Figure 9 shows the addressing structure of the 1108. The segment (called 'bank' in 1100 series manuals) select algorithm is given in Figure 10. The decision variable is a segment select value that determines which segment to use. Conceptually, each base register contains an 18-bit number, maintained by the Executive with the low-order 9-bits being 0. The value in a segment select register defines the length of the area based by Base 0 and is used to determine whether a relative address should be based by Base 0 or Base 1. An effective operand address for the 1107 is $u + X_m$; for the 1108, it is $u + X_m$ + base.

Since the base registers are 18 bits, a storage capability of 262K words is possible. Normally this space is occupied by the Executive and several programs being multiprogrammed. Each program can address up to 65K words without indexing, and up to 262K data words (minus instruction space) with indexing, since X_m is 18 bits long.

Storage is assigned and protected on a 512-word boundary. Two storage limits registers are associated with each base register. These registers identify the extremities of the addressing range allowed to each segment. Access outside the segments is not allowed and write protection is selectable within the segment.

The 1108 relative addressing structure provides a relocation capability to allow efficient multiprogramming. The incorporation of a privileged-mode and a better memory protection scheme provides the necessary isolation for multiprogramming. Instructions dealing with control of system resources, input/output instructions, for example, were reclassified as privileged. The requirement for multiprogramming led to the addition of a duplicate General Register Set similar to the one shown in Figure 3; one GRS is intended for the user and the other for the Executive. The user cannot use the Executive's GRS but the Executive can

Fig. 9. Address structure of 1108 architecture.

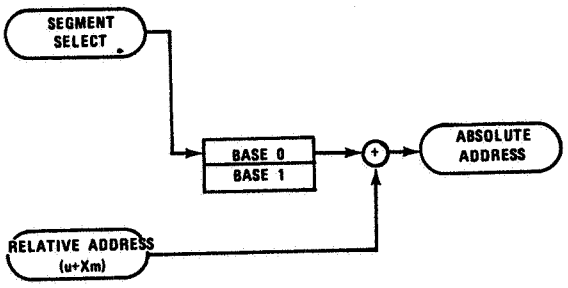
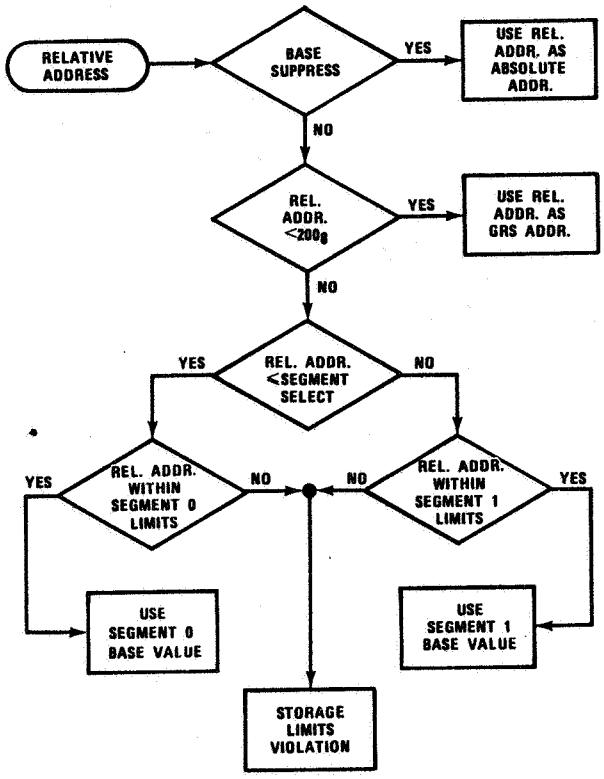


Fig. 10. Segment select algorithm for 1108 architecture.



use either set. This additional set of registers reduces the context switching time. To reduce context switching time even further when the Executive makes use of the user's GRS, the Executive normally uses (by software convention) a "minor set" of only 13 registers.

2.2.2 Multiprocessing

Later versions of the 1108 can be configured as tightly coupled, symmetric multiprocessors. This requires multiple access ports at each of the main storage units. The Test And Set instruction was added to facilitate interprocess synchronization. This instruction causes the storage unit to read a semaphore bit, and then, without allowing any other processor to access the same memory word, to set the semaphore bit. If the semaphore was initially set, an interrupt occurs (indicating that the item protected by this semaphore is already being used). At this point the interrupted process is queued until the semaphore is cleared. If the semaphore was initially clear, the next instruction is executed. Execution of the Test and Set instruction must precede the use of any data where erroneous results could be produced by two or more instruction streams operating on these data concurrently.

The introduction of the multiprocessor version of the 1108 led to the development of a new kind of system component, called the Availability Control Unit (ACU). This unit allows partitioning of the system into three smaller independent systems for debugging of either hardware or software on one system, while normal operation (at reduced throughput) continues on the remainder of the system. Each processor periodically sends a signal to the ACU indicating that the processor is still functioning and the Executive is still in control. If the ACU does not receive all the expected signals within a predetermined time, an automatic recovery sequence is initiated.

Some of the problems encountered in the expansion of the system from unit processor to multiprocessor configurations were: (1) The performance capability of main storage was strained by the increase in system activity, resulting in less-than-expected processor performance and increased frequency of i/o data overruns. (2) Some of the timing margins provided on the processor to main storage interface were reduced to accommodate the increased delays resulting from longer cables and multiple port switches. (3) The system interfaces and the grounding scheme were strained by the increased number of interfaces required and the larger physical size of the system.

2.2.3 Arithmetic

A set of four double-precision floating point instructions, and instructions to aid in conversion between integer and the new floating-point format were added to the 1107 repertoire. The range of floating-point numbers was extended to approximately 10^{308} to 10^{-308} and the precison was increased to 60 bits.

The format for double-precision, floating-point words is shown in Figure 7b. Double-precision operands are located in adjacent registers with the instruction specifying register A (the most significant half) and implying $A+1$ (the least significant half). There is also one instruction, a double shift and count, that places the number of shifts in $A+2$. Since the specified A register could be A_{17}, registers 34_8 and 35_8 must be reserved as shown on Figure 3. Double-precision operands can start on any word boundary in either GRS or main storage.

In addition to the double-precision, floating-point instructions, the 1108 provides double-precision integer add and integer subtract instructions, along with double-word register load, store, and test instructions.

2.2.4 Input/Output

The 1108 has up to 16 input/output channels integral to the processor. These channels are faster than, and upward compatible with, the 1107 channels. The

1108 also permits the inclusion of one or two separate input/output controllers; each controller contains up to 16 input/output channels.

2.2.5 Technology

The 1108 processor is constructed of discrete components except for the General Register Set, which is composed of integrated circuits. Main storage is core memory with a 750-nanosecond cycle time.

2.3 1106 System

The 1106 system was introduced in 1969 as a lower priced alternative to the 1108. The 1106 is architecturally similar to the 1108; the main differences are storage and timing changes. Two main storage units, both core, are available for the 1106. One of them has a cycle time of 1.5 microseconds and is referred to as "unitized" storage. This storage is designed in 131K word modules; consequently, instruction execution time is a function of storage size. If there are only 131K words of storage, it is not possible to perform an operand access and simultaneously prefetch the next instruction; thus, the basic add time is 3 microseconds. With 262K words of storage, memory overlap can be used to reduce the basic add time to 1.5 microseconds. The second, and faster, storage unit is constructed of 32K word modules. The system using this storage is called the 1106-II and the basic add time is 1.0 microsecond.

2.4 1110 System

The 1110 system, first delivered in 1972, was designed to improve upon the 1108 by increasing the degree of multiprogramming, improving the capability in a business environment, and allowing larger multiprocessing configurations. Changes from the 1108 include:

- named segment address space
  - 2^{12} segments in logical space
  - 2^{30} words in logical space
  - enhanced storage protection
  - physical address space expanded to 16M words
- new external i/o processing unit having provisions for data chaining and interrupt tabling
- additional input/output instructions to communicate with input/output unit
- 2-level storage with a maximum of 262K words primary and 1048K words extended storage
- breakpoint and jump-history instructions to aid debugging
- byte manipulation instructions
- the number of i/o units expanded to four
- the number of channels per input/output unit increased from 16 to 24
- faster instruction execution
- provisions for four processor configurations
- 4-deep instruction overlapping
- parity on the input/output channel interface

2.4.1 Addressing

A conceptual view of the 1110 addressing structure is given in Figure 11. This structure would directly implement a segmented virtual address space if the Segment Descriptor Index was provided as part of each instruction. Since no space exists in the 1100 series instruction words to hold this segment index, an approach has been provided on the 1110 that utilizes four high-speed segment descriptor registers to hold the descriptors for up to four currently active segments. This accelerated addressing method is depicted in Figure 12. The algorithm for selecting one of the accelerated segment descriptors is shown in Figure 13. The segment descriptor register is divided into two sets of two descriptors each. At any time, either segment descriptor set 0,1 or 2,3 is primary. The other set may be designated as primary by branching into a segment that currently is in the secondary set of the segment descriptor register. Within a set, the segment to be used is selected using segment select values (0,2; 1,3) in a manner similar to the 1108 segment selection scheme.

The user loads the segment descriptor register using nonprivileged instructions. These instructions obtain new segment descriptors from the segment descriptor table, which is maintained by the Executive. Since there are 4096 segment descriptors in the segment descriptor table, and since each segment can have a maximum length of 262K words, the user has a named-segment address space of one billion words with which to work. Segment descriptors can appear in more than one segment descriptor table. This allows processors to share common code. Extensive use of this capability by the operating system reduces the amount of main storage space allocated to redundant code.

The base values on the 1110 were increased in length from 18 to 24 bits to provide a larger addressable main storage capacity. To maintain an affordable price with a larger memory, it was decided to use 2-level storage. Physically, the largest configuration allows up to 262K words of fast, plated-wire storage (primary storage) and 1M words of slower, lower-cost, core memory (extended storage). The extended storage begins at absolute address 1,048,576. The design is such that the highest absolute address of primary storage is 1,048,575 regardless of the amount of storage a particular system has; thus, primary and extended storage addresses are always contiguous. The most common usage of the four accelerated segments is: instructions in primary; data in primary; instructions in extended; and data in extended.

Insofar as physical space allocation is concerned, a user may specify whether a segment is to be located in primary storage or extended storage. A user who is

Fig. 11. Logical address structure of 1110 architecture.

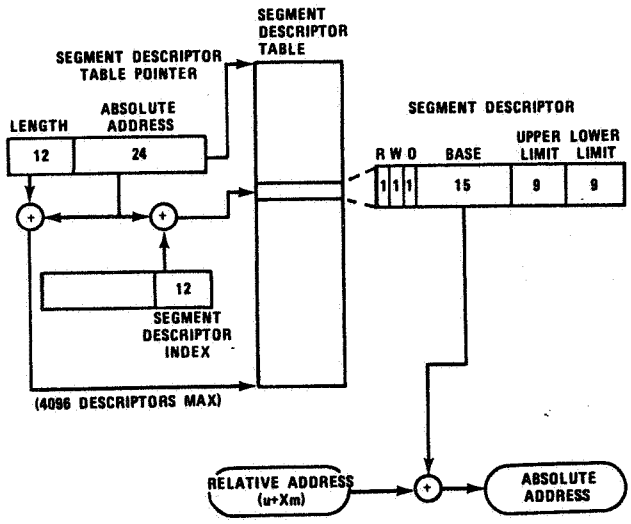
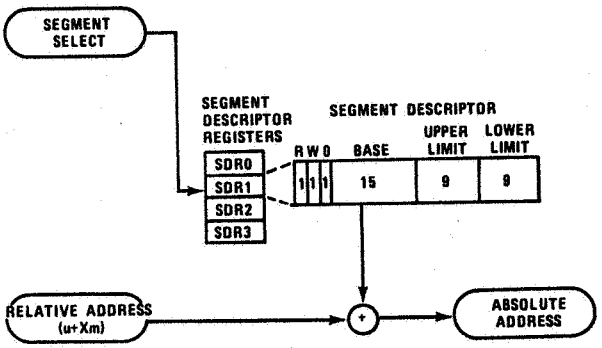


Fig. 12. Accelerated addressing structure for 1110 architecture.

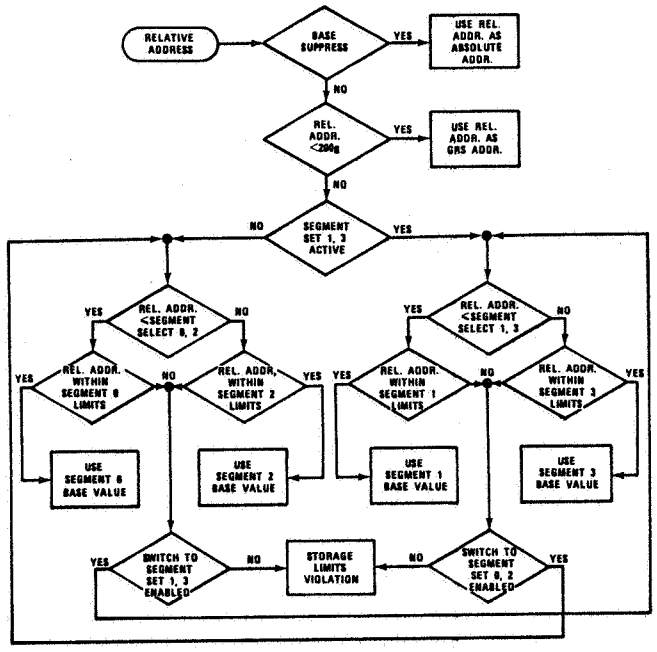


aware of the referencing patterns of his program, can make efficient use of the two speeds of storage. The Executive maintains storage reference counts to assist the user in proper program placement. Unless overridden by the user, The Executive will tend to place compute-bound programs in primary storage, and input/output-bound programs in extended storage. It has been found, by instrumentation, that about 90 percent of the main storage references within the Executive are made to 10 percent of the addresses. As a result, the Executive is partitioned to use both primary and extended storage. A new privileged state was added to permit the Executive to address all of physical main storage without the use of base registers by defining the X_m portion of an index register to be 24 bits and the X_i portion to be 12 bits.

2.4.2 Byte-Handling Instructions

A set of 24 instructions for byte handling were added to the base 1108 repertoire to improve the capability of the system for business applications. A sampling of these instructions includes: Byte Move, Byte Move With Translate, Byte Compare, Byte To Packed Decimal Conversion, Byte Add, Byte Subtract, and Edit. Instructions such as the Byte Move operate on bytes of 6, 9, 12, or 18 bits. These instructions use some of the previously unassigned (on the 1107 and

Fig. 13. Segment select algorithm for 1110 architecture.



1108) R registers to parameterize the byte-handling instructions; for example, they define the number of bits per byte, number of bytes in a string, and special editing characters.

2.4.3 Overlap

To achieve the desired basic add time of 300 nanoseconds on the 1110, it was necessary to use a high degree of instruction execution overlap. A 4-deep instruction stack provides the nucleus for the overlap. This complicated the design because of the conflict checks that must be performed. For example, if a register used for indexing on instruction $n + 1$ has its contents changed by instruction n, the operand fetch for instruction $n + 1$ must be delayed. This can degrade performance; however, most of the compilers generate stack optimized instructions. In addition, many of the assembly language instruction sequences within the operating system have likewise been optimized.

2.4.4 Technology

The 1110 was the first processor in the 1100 series to be constructed entirely of integrated circuits; in particular, high-speed TTL is used. Primary storage is composed of plated wire with a 280-nanosecond read cycle time and a 480-nanosecond write cycle time; extended storage is core storage with a 1.5-microsecond read/write cycle time.

2.5 1100/20, 1100/10 Systems

The 1100/20 and 1100/10, successors to the 1108 and 1106, were first delivered in 1975 and 1976,

respectively. These systems have the same instruction set as the 1108 and 1106.

The storage capacity on the 1108 was found to be inadequate for many applications; furthermore, semiconductor technology has lowered storage costs since the introduction of the 1108. As a result, modifications were made to the 1108 addressing structure to increase the addressable storage range. The storage addressing range was increased to 524K words by appending a state bit to an 18-bit address—produced as in the 1108—to yield a 19-bit address. The logical address space is still limited to 262K words, but the additional storage is usable by allowing more programs to be multiprogrammed. Additionally, the input/output control word (Figure 8) was redefined so that the address field contains 19 rather than 18 bits. Consequently, the count field was reduced in length by one bit. This format change has no impact on the user since the Executive has always formatted these control words for the user.

2.5.1 Technology

Both models use MOS main storage with single-bit error correction and double-bit error detection. The 1100/20 has a storage cycle time of 450 nanoseconds and a basic add time of 875 nanoseconds; the 1100/10 has a storage cycle time of 650 nanoseconds and a basic add time of 1125 nanoseconds. The memory cycle times are less than the minimum instruction execution times in order to provide more efficient multiprocessing operation.

2.6 1100/40 System

The 1100/40 system, which is architecturally similar to the 1110, was first delivered in 1975. The instruction execution times for the 1100/40 are the same as those for the 1110 when executing out of primary storage. However, two changes from the 1110 allow greater overall performance. Extended storage is constructed from MOS memory chips and has a read/write cycle time of 800 nanoseconds. Single-bit error correction and double-bit error detection are incorporated. The faster cycle time allows faster execution of instructions and data located in extended storage. The other change was to construct primary storage out of bipolar memory chips instead of plated wire. The 1110 is constrained to 262K words of primary storage by cable length, which is a function of storage unit size. Long interface cables to memory preclude the short access time needed on primary storage. The more compact semiconductor storage allows the configuring of 524K words of primary storage on the 1100/40.

2.7 1100/80 System

The 1100/80, with a basic add time of 200 nanoseconds, is the latest and most powerful member of the 1100 series family. It makes an interesting contrast with the 1107. The 1107 had a General Register Set that operated at 667 nanoseconds/cycle, and several cycles were necessary to execute an instruction. The 1100/80 completes the entire instruction in only 200 nanoseconds. The first delivery of the 1100/80 was made in 1977.

The 1100/80 system introduced several improvements to the 1100 architecture and design approach. The most significant are:

- addressable memory returned to a single level structure
- memory addressing structure simplified
- the effective speed of memory increased by use of a user-transparent cache memory
- single-bit-error correction and double-bit error detection on main memory
- arithmetic/logic unit microprogrammed
- emulator provided for executing Sperry Univac 494 programs
- instructions added to accelerate user and Executive common functions
- higher throughput achieved by using faster logic components and more dense packaging
- automatic recovery from system failures improved

A new group of instructions has been added to accelerate common functions for both users and the Executive. These include several context switching instructions, such as save and restore system status, and load and store GRS; and user-oriented instructions, including new constant storage and memory increment and decrement instructions.

Two new instructions were also added to support the autorecovery feature of the 1100/80. These instructions reset the autorecovery timer and toggle the autorecovery path. When autorecovery is enabled, and the system software does not reset the automatic recovery timer within the preset time interval, the System Transition Unit (similar to the Availability Control Unit of the 1108) clears, reloads, and reinitiates the system. Two recovery paths are provided. The alternative recovery path is system initiated when an attempted automatic recovery fails. The instructions mentioned above provide for software resetting of the automatic recovery timer and for selection of the first automatic recovery path to be used by the next recovery attempt.

2.7.1 Addressing

The 1100/80 logical address structure, shown in Figure 14, is similar to that of the 1110. The accelerated addressing structure, shown in Figure 15, is also similar to that of the 1110. The significant difference in the addressing structures of the 1100/80 and 1110 lies in the segment select algorithms. A comparison of the 1100/80 segment select algorithm, shown in Figure 16, with that of the 1110 (Figure 13) shows that the 1100/80 algorithm is simpler. The 1100/80 addressing

Fig. 14. Logical address structure of 1100/80 architecture.

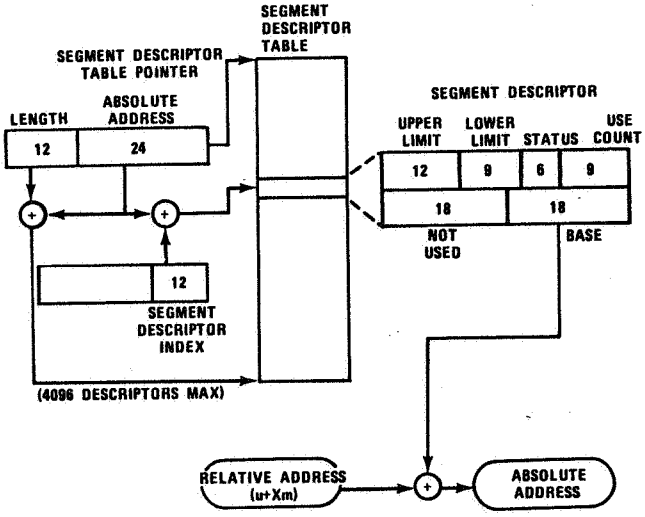
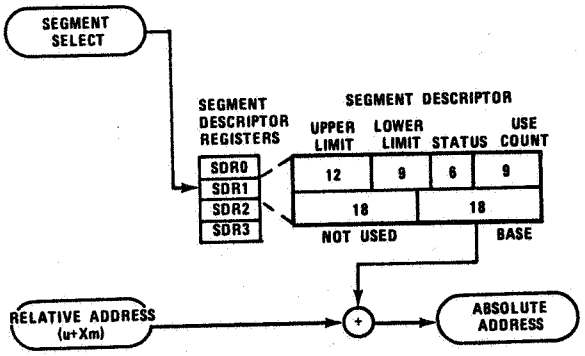


Fig. 15. Accelerated addressing structure for 1100/80 architecture.



algorithm has a simpler autoswitch function, and does not have the segment select variables of the 1110. Once a segment set is selected, the segment to be used is found by an ordered testing of the segment limits against the relative address. If no test succeeds in the primary set, the secondary set is tested.

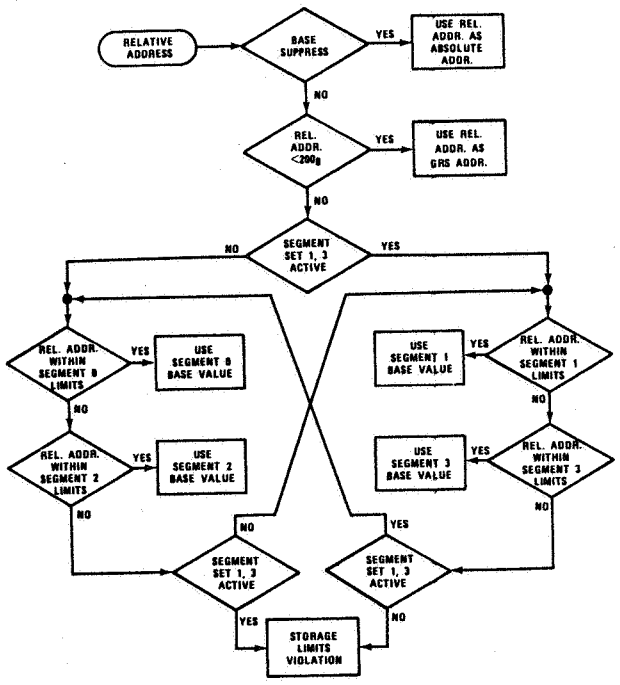
In addition to the simpler algorithm for selecting the next segment, the 1100/80 also introduced instructions to aid address-space manipulation. The most significant new instruction is one that transfers a 2-word segment descriptor directly from the segment descriptor table to the segment descriptor register, saves the previous contents of the segment descriptor register, and branches.

The granularity of segment sizes has been improved on the 1100/80. Segments can be as large as 262K words and can be specified in 64-word granules beginning on any 512-word boundary and ending on any 64-word boundary. Protection is also provided on this granularity.

2.7.2 Dual Mode

The 1100/80 system has an alternative, 30-bit word mode. In 30-bit mode, Sperry Univac 494 System user instructions are executed directly by the 494 system execution module of the 1100/80 system, or via soft-

Fig. 16. Segment select algorithm for 1100/80 architecture.



ware assistance provided by the Promega processor. Promega is an 1100 series user level program that provides a 494 virtual processor capability on the 1100/80 system for the purpose of running a 494 operating system. If, for example, the standard 494 operating system (Omega) is to be run, it is loaded as a user program under the 1100 series Executive. Promega provides the interface between Omega and the 1100 series operating system that allows Omega to be executed as a user program. Promega runs concurrently with other 1100 series user programs providing multiprogramming of mixed 1100 series and 494 series programs.

Each 494 system instruction is emulated either by the 1100/80 system hardware or by the Promega software. Generally, user instructions are emulated by hardware, and privileged instructions by software. Promega allows Omega to allocate system resources, as in a 494 system, to provide for multiprogramming of user jobs. The capability of Promega-Omega provides compatibility with the batch utility packages, application packages, and language processors of 494 Omega. Promega simulates the functional capabilities of the 494 system instruction processor, including clocking, interrupt generation and handling, and execution of privileged instructions. The 494 system user instructions are executed directly by hardware and thus are transparent to Promega. The privileged instructions, including input/output requests, are passed to Promega via an emulation interrupt and executed by the software.

2.7.3 Input/Output

Input/output channels on the 1100/80 are available in two forms. Word channels are available that are compatible with the 1110 system. Additionally, intelli-

gent byte channels are available that allow the direct usage of byte oriented peripheral equipment, whereas previously these required separate adapters. As in the 1110, the input/output channels are housed in their own separate cabinet.

2.7.4 Technology

The 1100/80 uses a high-speed cache memory between the processor and main storage. The cache memory is transparent to the user. It is constructed of emitter-coupled logic storage elements and contains up to 16K *words;* these words are the most recently used contents of main storage. The cache has an access time of 120 nanoseconds.

The physical main storage capacity was increased to a maximum of 4M words of MOS memory with a read access time of 1250 nanoseconds. Single-bit error correction and double-bit error detection are provided.

The processor is constructed of emitter-coupled logic integrated circuits, necessitating a controlled impedance signal distribution system. Consequently, the backpanel wiring is accomplished with four large multilayer printed wiring boards. The ICs themselves are mounted on smaller (7 inch by 10 inch) multilayer boards. The predecessor systems used wire wrap backpanels.

2.8 Future Architecture and Technology

There are two main driving forces which, together, will shape the form of the evolution of the 1100 series. One of these forces is user requirements; the other is the available technology. The time between the commencement of a system design and first delivery to users can be very long, often more than five years. The time between start of design and required availability of the logic elements in substantial quantities can be as long as two years. Since both user requirements and the best available technology vary with time, it is difficult to predict the best possible computer design. As a result, it is often necessary to carry out multiple design efforts and to decide later which of these systems should actually be introduced as products.

The area of user requirements is difficult to assess. Traditionally, short term throughput per dollar (usually determined by short benchmark runs) has been the main metric by which systems have been evaluated. However, it seems clear that other aspects of computer systems will be receiving increasing emphasis by the user community. Primary among these are expected to be security, availability, system integrity, programmer productivity, and ease of use. The relative emphasis on each of these aspects will vary from user to user, making it difficult to predict what the overall profile will be. It is not clear how much users will be willing to pay to achieve these more subjective attributes. Some of these attributes, such as security, will be achieved only at the expense of throughput per unit cost. Conversely, some of the other attributes, such as increased system availability and increased programmer productivity, can lead to an increase in overall throughput per total (purchase plus support) cost. However, if system value continues to be measured on short-term throughput per unit cost, it may be difficult to sell those systems that actually have better overall throughput per total cost. It is because of the difficulties of accurately predicting technological capabilities and user requirements that multiple options are currently being pursued with respect to future 1100 systems.

Three possible future 1100 systems are discussed in the following text. Although all are being actively pursued, no ordering is implied as to the relative priority or relative level of activity on any of these projects. No implication is being made that these are the only options that are being pursued, or that any of these options will ever appear as future Sperry Univac products.

One option being pursued is the utilization of large-scale integration in the design of an 1100 processor. Although LSI has been used extensively in the design of memory systems, very little use of LSI has been made so far in the design of mainframe level processors. The approach being used is to utilize multiple microprocessors to create an 1100 system processor. It has been demonstrated that mainframe level performance (greater than 1108 performance) can be achieved by appropriately interconnecting sets of microprocessor units. We believe this to be the first example of achieving mainframe level performance on a single instruction stream using a single microprocessor slice as the basic building block. This performance is achieved by decomposing macroinstructions into a set of activities which can be executed in parallel. These parallel activities are executed by a set of microinstruction streams which are, in turn, built up from sets of microprocessor slices. By appropriately adjusting the interconnections, a varying number of microlevel paths of varying path widths can be established. The very low cost of the basic building blocks makes it possible to economically perform some parallel activity on each of the microinstruction streams. This parallel activity allows partial execution of both possible next instructions on conditional branches, thus allowing overlapped microinstruction streams with negligible degradation due to conditional branching. Furthermore, because the basic LSI microprocessor building blocks will be relatively inexpensive compared to the rest of the system, very high system integrity can be achieved via complete duplication and comparison of the microprocessor results. Parity checking is effective in the control logic because a table-driven control structure is used. These fault-detection schemes create a processor design where the probability of detecting any fault is about 0.9 at a 15 percent increment to the processor build cost. A beneficial fallout of this design approach is that the resultant 1100 system is microprogrammed.

The 1100/80 is currently the highest performance 1100 system. Several possibilities exist for improving upon this performance, including a change to a higher performance technology, more sophisticated pipelining approaches, and the addition of certain new instructions. The 1100/80 logic elements have an average gate delay of about 2 nanoseconds. Subnanosecond emitter-coupled logic is now available with higher levels of integration than that used with the 1100/80. Higher packaging densities are also available. As the 1100 series addressing structure has evolved to create an expanded environment for the user, the implementation difficulties have increased. Several possible permutations of overlapped address calculation, coupled with parallel or pipelined cache systems, have been identified and evaluated to overcome these difficulties. Similarly, new byte and array manipulation instructions have been identified and evaluated. It is believed that a significant performance improvement can be achieved by utilizing these techniques.

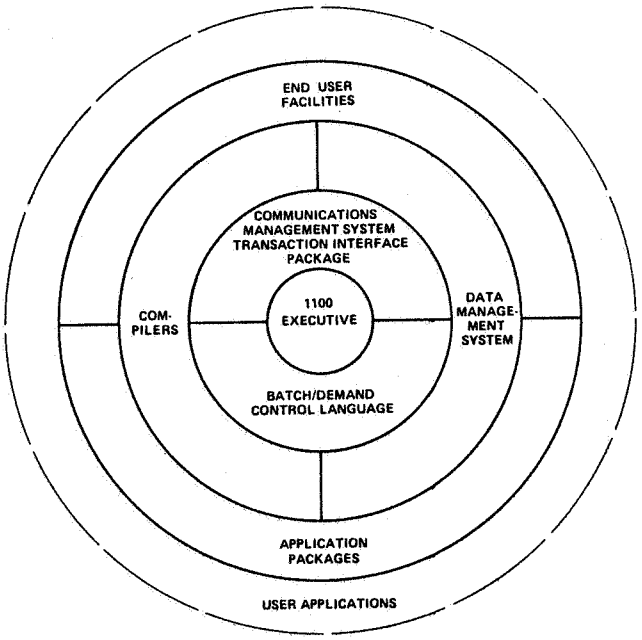
The two approaches described above are mainly directed toward providing improvements in cost/performance and integrity. There is also an activity under way that is directed toward achieving significant improvements in other areas by providing an alternative mode of operation. The architecture of the new mode is substantially different from that of the 1100 series. Therefore, it is necessary to switch back to the old mode to execute 1100 series programs. This design is based on a general emulator which supports an efficient virtual machine structure capable of executing multiple copies of both old- and new-mode systems. The incorporation of a new mode makes it possible to utilize very sophisticated architectural concepts to create strong capabilities in several important areas, such as security, efficient execution of selected higher-level languages, and programmer productivity. The establishment of a dual mode of operation makes it possible to execute existing 1100 series programs, while exploiting the architecturally enhanced environment for those applications that require increased capability. It is anticipated that the efficient underlying emulation structure, coupled with the decreasing cost of hardware, will make this dual mode of operation cost effective.

Some possibilities for future 1100 systems have been discussed. These projected systems will all execute existing 1100 series programs. The systems that actually emerge as products will be the ones that best anticipated the strongest requirements of potential users, and which best exploit the evolving technology.

3. Operating System

The 1100 series Operating System consists of a series of 'layers' of software as illustrated in Figure 17. To be discussed, in turn, are: the control portion or Executive (analogous terms used in the industry are

Fig. 17. 1100 Operating system.



Supervisor, Monitor, Operating System and Master Control Program), including the control language; the languages; and the database and data communications software.

3.1 Executive

The first 1100 series executive system, EXEC I, was introduced on the 1107 in 1962. It supported a general purpose multiprogramming environment. Functionally, EXEC I included many of the facilities of current executive systems, such as priority based job scheduling, logical facility assignments, relative address program loading, and control switching between concurrent programs based on time quantum or loss of control. The major exceptions were the lack of a workspace-oriented file facility, and an easy to use control language. The lack of these facilities posed a problem which was difficult to correct short of complete rewrite, since the control language and file management philosophies were integral to the basic design of EXEC I.

In early 1963 a second executive, EXEC II, was released. EXEC II became the more widely used 1107 Executive (both EXEC I and EXEC II were also used at the early 1108 sites). Although developed concurrently with EXEC I, EXEC II did benefit from the early EXEC I experience and contained a much more flexible, easy to use control language and a workspace-oriented program and data file facility. It also introduced an input/output control routine concept called a Symbiont (spooling routine). These routines overlapped read/print/punch operations with program execution. However, multiprogramming of user programs was not provided. Each program executed serially.

Fig. 18. 1100 Series Control Language.

```
@RUN JONES,421706
    (Identifies user and account number)
@COB,I PROG1, RB1
    IDENTIFICATION DIVISION
    • (Compiles a COBOL symbolic element into a relocatable
      element)
    • (The source images may follow in the run stream or be added
      from a mass storage file)
    •
    •
@COB,I PROG2,RB2
    IDENTIFICATION DIVISION
    • (Compiles a second COBOL symbolic element)
    •
    •
@MAP ,ABS
  IN RB1
  IN RB2
    (Collects two or more relocatable elements to form an absolute
      element)
@XQT ABS
    (Loads and executes the absolute element which accesses a data
      base and produces a report)
@FIN
    (initiates run stream termination)
```

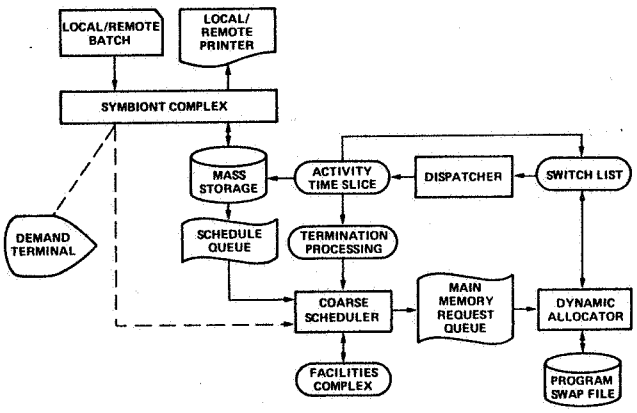
Note: Note all Control Language statements are illustrated. This run stream may be used in either batch or demand modes. An interactive text editor is provided for symbolic data manipulation, for correction of compiler input, data preparation and performing run-stream editing.

Despite their many capabilities both EXEC I and EXEC II shared a common deficiency: both were completely batch oriented when timesharing was beginning to appear on the scene. This was a problem not easily corrected without a complete rewrite; consequently, the decision was made to develop a totally new executive, this time in conjunction with the introduction of the 1108 system.

This new executive (often referred to as 'EXEC 8') for the 1108 was first released in 1967. It combined the multiprogramming capabilities of EXEC I with the control language, program and data files, and symbiont capabilities of EXEC II. Designed to operate in both a multiprogram and multiprocessor environment, the 1100 Executive System, as it is now called, supports a concurrent mix of batch, remote batch, demand, and transaction programs. Particular emphasis was placed on demand mode (timesharing). All system facilities available in batch mode are likewise available in demand mode; the same run stream (Figure 18) can be used in either batch mode or demand mode without change. This continues to be one of the strongest features of the control language. The Executive also has a transaction mode. This mode is intended for realtime environments such as an airline reservation or bank teller system. A transaction is an input processed by a user-written program and which typically results in an output.

The following paragraphs define some of the major terms and system concepts used by the 1100 Executive. (Figure 18 illustrates the use of the control language. Figure 19 illustrates the flow through the Executive.)

Fig. 19. 1100 Series executive.



A run stream is a collection of user service requests used for batch, remote batch, and demand. Included in the run stream are any data images supporting each service request. The run stream is formulated using a control language consisting of service requests. A task is an individual service request; for example, assign a tape, or perform a compilation within a run stream. The execution of the run stream is referred to as a 'run'.

A program has its origins in symbolic elements within the run stream, which are then compiled to form relocatable elements. The elements are, in turn, collected (bound) with other relocatable elements to form an absolute element (the program). The term "absolute" refers to program relative address resolution only; the relative addressing capability of the hardware allows the program to be loaded (or swapped and reloaded) and executed anywhere within main storage. References to shared segments of both user code and system libraries are resolved during execution.

When a batch run stream enters the system, it is first processed by the symbiont complex, which disassociates the run stream from the relatively slow unit-record device speeds, and allows tasks to proceed at higher mass storage speeds. The run stream is scanned for facility allocation and prescheduling. The symbionts provide multiple, asynchronous, unit record input/output; this is particularly important in a multiprogramming/multiprocessing environment.

After staging in a mass storage schedule queue, the run stream is processed by the Coarse Scheduler which is responsible for the scheduling, selecting, and activating of the run stream. The run is assigned a Temporary Program File, or workspace, in which intermediate results such as compiler output or text editor output are held until termination, when the file is released. The results may be retained by directing them instead to a permanent file or copying the contents of the temporary file prior to termination. Demand runs are scheduled for immediate activation; batch runs are scheduled on a user-selected priority basis. A major batch scheduling consideration at this point is the

availability of facilities such as tape devices. When the necessary facilities are available the program is then queued for main storage allocation.

The program is removed from the main storage allocation queue by the Dynamic Allocator, which has the responsibility for the distribution of main storage space among users, on a priority basis. The main functions of the Dynamic Allocator include the allocation and release of main storage and the initial load, swap-out, and reload of programs. The decisions of the Dynamic Allocator are facilitated by relative addressing which allows the program to be loaded anywhere in memory. The program may be partitioned into multiple segments, which need not be loaded contiguously. Segments (usually shared) that were not a part of the initial program may be referenced (and loaded) dynamically. This facility allows for greater program protection and re-entrancy efficiency. It also allows each segment to be loaded into a separate memory module, thus reducing the effective instruction execution time through storage unit overlap.

The 1100 Executive uses a switch list to control an executing program. The program may fork into any number of asynchronous execution paths, called activities, each of which is independently scheduled (except for synchronization requests). It is given main memory space and a time slice under the control of a unique switch list entry. The executing activity requests 1100 Executive services through a set of executive requests. These perform services such as input/output, control statement processing, and activity activation.

Real processor time slices are allocated to outstanding activities by the dispatcher according to the priority and needs of the respective activities. In a multiprocessor system, any available processor may be assigned to execute the next time slice. Thus, an activity may execute successive time slices on different processors, or two parallel activities within the same program may be executing concurrently on different processors. Upon completion of processing within an activity, a request for normal or abnormal termination is made. After all activities within a task have terminated, the task is terminated and control is returned to the Coarse Scheduler. When the final control statement is detected, the run is closed and run termination accounting information is written to a system log.

The initial release of the 1100 Executive was made in early 1967. Typical of newly introduced system software, a rapid progression of stability and minor enhancement releases quickly followed. The major emphasis was on stability as the development project grappled with the problems of debugging the Executive for a demand environment. Debugging difficulties resulted from handling many simultaneous terminal sessions instead of repeatable batch programs. Considerable effort was expended in tuning the resource allocation algorithms used by the Dynamic Allocator and Dispatcher to control the sharing of resources between demand programs. The original timesharing algorithms, based on processor utilization alone, proved inadequate and caused excessive program swapping. Improved algorithms were developed that incorporated input/output operations and program size into the determination of the time slice allocated to each program.

Despite these initial growing pains, the 1100 Executive, measured by the test of time, has proven to be of solid design. Over the years many routines and algorithms have been redesigned. Many new peripheral handling routines have been added. However, the basic structure or foundation described above remains intact. Part of the credit for this is due to the Programmer Reference Manual, which was written before implementation and served as the design document and implementation control vehicle.

The progressive introduction of the subsequent members of the 1100 series provided the most significant test of the basic logical structure of the 1100 Executive. Input/output control, program swapping, interrupt processing, and other time-dependent routines had to be adjusted, first for slower machines such as the 1106, then for faster machines such as the 1110 and 1100/80. A basic design goal, that of maintaining a single source (symbolic) of all components of the operating system, proved to be a key factor in providing a uniform environment in all systems. The symbolics are released to all sites. System generation tools are provided which allow a site selectively to include or exclude certain functions. These tools have been extended, particularly in the case of the 1100 Executive, to similarly include or exclude model-dependent code. Special care has been taken to isolate model-dependent instructions whenever possible. Thus, the same symbolic version of the 1100 Executive can be used to generate a code which will execute on all post-1107 computers. Despite considerable architectural differences between the various 1100 systems, only about 10 percent of the 1100 Executive is hardware dependent. Given the enormous cost of developing a new executive system or of maintaining several, it is doubtful that the 1100 series product line could be as large and diverse as it is today were it not for the single symbolic concept.

The 1100 Executive insulates much of the remaining operating system software and all user software from the physical characteristics of the computer. For example, in the file management area, files are assigned with a symbolic name to a logical device class. The file may migrate from drum to fixed disk, to removable disk, without requiring any change in the accessing program. If inactive for a long period, the 1100 Executive may transfer a file to tape and subsequently transfer it back to mass storage upon the next reference. Except for possible time delays, such movement of files within the virtual file structure is transparent to the user. Each file is viewed as a logical address space. All access is file relative (number of words to read/write and offset into the file), regardless of physi-

cal track size and formatting factors. All user access to symbolic, relocatable, and absolute elements within a program file is made symbolically by element name without reference to logical addresses. User access to information within a data file is based upon logical address unless a facility such as the Data Management System is being used to provide symbolic access. Files may be preallocated as contiguous physical entities or incrementally allocated as needed to noncontiguous physical space. Such insulation from the hardware has been a major benefit to the 'outer layers' of the operating system software. The compilers and database software utilize the standard 1100 Executive facilities and interfaces. These have only isolated hardware-dependent code. For example, the compilers make use of this code for generating the newer instructions introduced with the 1110 and 1100/80.

3.2 Languages

The operating system software components for each language consist of the compiler and a set of associated run time library routines. The original compilers for Cobol, Fortran IV and Fortran V date back to the 1107 and were based on a 6-bit character set. Their replacements, discussed below, have adopted the 8-bit ASCII character set. Table II summarizes the characteristics of the major ASCII compilers. These compilers, for Cobol, Fortran, PL/I, and a compiler for Sperry Univac use only (PLUS), are briefly described below. Also available are Algol and Basic compilers, an APL interpreter, an assembler, and a meta assembler.

The ASCII-based Cobol compiler was a major step in the evolution of the 1100 series from a scientific to a business-oriented system. It demonstrates that an efficient and competitive business compiler can be implemented using the basic 1100 instruction set. In addition, the compiler takes advantage of the byte instructions added to the 1110 hardware. These byte instructions were coupled with string manipulation capability within the compiler to further enhance its business orientation. The initial release included a Codasyl Report Writer, Message Control System, and Asynchronous Processing System as extensions to the then current Cobol standard. The compiler has been upgraded to comply with the latest ANSI standard, which became a federal government requirement during 1977.

The ASCII-based Fortran compiler and the PL/I compiler are noteworthy from an implementation viewpoint: first, they are implemented in a higher-level language; and second, they are implemented as part of a "common compiler model". The higher-level implementation language is PLUS (Programming Language for Univac Systems). PLUS was defined and developed for the express purpose of implementing new software in a common language on both the 1100 and 90 series. (The Sperry Univac Series 90 is a small-to-medium scale product line based on a byte-oriented architecture.) The compiler is intended to be an internal development tool, not a customer product. Unlike some higher level languages designed for internal development purposes, PLUS language source code cannot be intermixed with assembly language source code within the same symbolic element. The PLUS compiler is thus free to perform extensive optimization without regard to the possibility of assembly language manipulation of registers or storage contents. The systems programmer is not given the option of reverting to assembly language in the same routine, which would limit the portability of the code to other Sperry Univac systems. As a result, PLUS is generally used for new products.

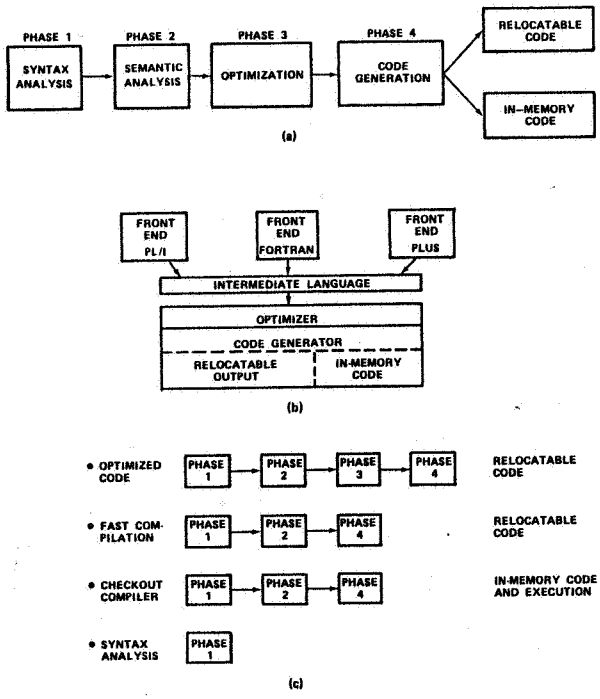
Table II. Characteristics of the Major[1] Compilers

	Cobol	Fortran	PL/I	PLUS
Initial release	1972[2]	1975[2]	1975	1975
Latest release	1977	1977	1977	1977
Re-entrant compiler	Yes	Yes	Yes	Yes
Re-entrant output	Yes	Yes	Yes	Yes
Demand-mode syntax scanning	Yes	Yes	No	No
Checkout mode	No	Yes	Yes	No
Global optimization	Yes	Yes	Yes	Yes
TIP interface	Yes	Yes	No	No
DMS interface	Yes	Yes	Yes	No
Standards	Yes[3]	Yes[4]	Yes[5]	No

[1] Other languages available include Basic, APL, Algol and Assembler
[2] Total replacements for earlier compilers
[3] ANSI X3.23-1974, FIPS PUB 21-1
[4] ANSI X3.9-1966 (and portions of proposed 1977 standard)
[5] ANSI X3.53-1976

In addition to being implemented in PLUS, the Fortran and PL/I (and PLUS) compilers are implemented under a design referred to as the "common compiler model." The common compiler model utilizes a 4-phase approach. Figure 20a shows these four phases. The first phase is syntax analysis. This is performed by a separate front end for each language. The second phase is semantic analysis. This phase results in the generation of intermediate text. The third phase is optimization, which optimizes the intermediate text. The fourth phase is code generation which generates either relocatable code (to be collected to form a normal program) or in-memory code (to be immediately executed for checkout purposes). All four phases are reentrant. As shown by Figure 20b, the latter three phases are implemented using a common design architecture for Fortran, PL/I, and PLUS. In addition to allowing partial commonality across programming languages, this modularization creates other benefits. Figure 20c illustrates how Phase 1 may be used to perform syntax analysis for text editing purposes. Phases 1, 2, and 4 (generating relocatable code) may be used as a high-speed compiler. Phases 1, 2, and 4 (generating in-memory code) may be used as a checkout compiler which allows the user to trace the execution, halt the execution, dump variable values, and perform other debugging activities. Finally, all four phases may be

Fig. 20. Common compiler model.



used to generate optimized relocatable code for a production program.

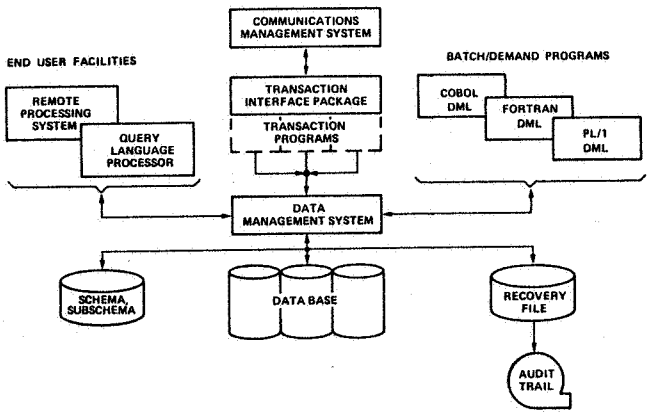
Fortran and PL/I meet and extend the full ANSI standard. A common run-time re-entrant library is shared by all executing Fortran and PL/I programs.

A general purpose macro facility was developed in parallel with the Fortran and PL/I compilers. Macro consists of two components; the first supports the definition of macros; the second, callable by the Fortran and PL/I compilers, expands the macros into source text as part of the user program. With this facility, the Fortran and PL/I languages become extensible. For example, the macro facility has been used to provide data manipulation language extensions to Fortran.

3.3 Database and Data Communications

In late 1971, the 1100 series entered the emerging world of database management with the introduction of a Data Management System, DMS 1100, which is depicted in the lower portion of Figure 21. DMS 1100 is based on the specifications of the Codasyl Data Base Task Group (DBTG) and successor committees. It includes a separate data definition language for defining the characteristics of the database (the schema) and the security and validation requirements. A second data definition language defines a logical subset (the subschema) which consists of the items of information available to a specific programmer or end user. Data manipulation language commands are available as extensions to Cobol, Fortran, and PL/I. The commands are functionally equivalent across the host languages; however, the syntax used has a Cobol, Fortran, or PL/I style, as appropriate.

Fig. 21. Database and data communication software.



The online Data Management Routine supports a collection of alternative storage structures and access techniques. Records may be maintained on the database in one or more of six storage structures: direct, key transformation, index sequential, chained set, pointer array, and indexed pointer array. Records may be processed either randomly or sequentially, based on either physical (database key) or logical (symbolic key) criteria.

The Data Management Routine is multithread, allowing concurrent access to a shared database by batch, demand, and transaction programs. Conflict situations between concurrent programs are handled through the use of locks on the input/output unit, which typically contains several data records, plus automatic queuing and dequeuing mechanisms, and automatic deadlock detection and program rollback. Database integrity is provided by logging before and/or after images of altered pages to an audit trail tape and before images to a mass storage file. The audit trail tape images are used to recover the database in the event of a device failure. The mass storage images are used to rollback an executing program's updates in the event of a program error or system detected deadlock. The mass storage images are also used to rollback all executing programs in the event of a system failure.

A data maintenance utility provides a set of terminal-oriented privileged functions to be used by the data administrator to monitor and maintain the data base in an operational state. A second utility, the data reorganization utility, provides the data administrator with the capability to reorganize an existing database. It includes a reorganization control language used to indicate the portions of the database to be reorganized and the manner in which the reorganization is to proceed.

The upper portion of Figure 21 shows the communication and transaction software. A transaction interface package (TIP 1100) provides the generalized framework within which user transaction programs may operate at a real-time level. TIP 1100 augments the normal 1100 Executive mechanisms to provide

transaction program service functions, such as program loading and initiation. TIP 1100 includes a more specialized file control facility which has a shorter instruction path for transaction data. TIP 1100 files, which are 1100 Executive compatible, may contain a database managed by DMS 1100 as illustrated in Figure 21. This file control facility is also used for very high volume non-DMS 1100 environments such as airline reservation systems.

The communication management system (CMS) provides for the control and polling of the terminal network, the receipt and transmission of messages, the queuing of input messages for subsequent processing, and the transmission of output messages. It is the primary interface between the 1100 Executive (as a host) and any attached front-end processor.

There are two database oriented, end-user facilities. The first, the query language processor (QLP 1100), provides a conversational language interface for database retrieval and updating. The second, the remote processing system (RPS 1100), provides a screen (CRT) image interactive facility for data manipulation. These interfaces into DMS 1100 are illustrated in Figure 21.

QLP 1100 includes English language-like commands which require only identification of the items of information in the database necessary to satisfy the query. A simple query might be: LIST EMPLOYEE-NAME, EMPLOYEE-NUMBER, EXTENSION WHERE DEPARTMENT EQUALS MARKETING AND DEGREE EQUALS MBA. Knowledge of the physical structure of the database is not required. The DMS 1100 subschema data definition language gives the database administrator a means of controlling user access to the database. QLP 1100 contains a report writer which can be activated by a single command within a query session.

Unlike QLP 1100, which is command-language oriented, RPS 1100 provides a screen-image oriented interface to files maintained within a database. RPS 1100 allows the end user to view a file, such as a report on the status of all outstanding orders, in a relational manner. The user may advance images of data through the file viewing each screen, much like reviewing printed output. A line may be changed by typing over it, new lines may be inserted or old lines deleted. Computations, either horizontally or vertically, may be performed. Files can be searched, sorted, or matched with other files to produce a result file. Selected data from a file may be projected into a new file, or data from multiple files may be joined into a single file.

RPS 1100 includes an application development tool, known as the tutorial processor, which is dialogue-oriented, allowing an analyst to establish a sequence of operations tailored to an application. The analyst, instead of developing program specifications for subsequent coding by a programmer, interactively defines the application directly to RPS 1100 using the tutorial processor. This definition includes tailored screen image inputs, outputs, decision and branching criteria, and computational requirements. The end user is thus presented with screen images, which contain application-dependent, not system-dependent, terminology and style (e.g., hospital admission, order entry, customer service).

In addition to the two database oriented, end-user facilities, an end-user facility for program development is also provided. This facility, the high volume timesharing system (HVTS), utilizes the transaction interface package and communication management system and is designed to run from 50 to several hundred terminals in a timesharing environment. HVTS supports Fortran and Basic syntax analyzers and compilers, and an APL interpreter.

3.4 Future Software Directions

A number of software trends can be readily identified. Many of them have been underway for some time and are not unique to the 1100 series. Examples include the: increasing use of high level languages, movement toward online data base systems, growth in available application packages, emphasis on a programming methodology (structured programming), improvement in conversational programming techniques, increase in the level of system security, and development of additional industry standards. The operating system software discussed above—the Executive, languages, and database and data communication software—will continue to evolve in anticipation of and reaction to these trends. A sampling of several interesting development activities is discussed below. However, new development in the sense of new software packages or major additions to existing packages is only a portion of the development picture. The much less visible portion is the ongoing work to support new hardware additions (within the Executive) and maintain operational compatibility between the various releases of the operating system components discussed previously. Also ongoing are continual efforts to improve interfaces between components, reduce path lengths, and improve the program scheduling/swapping and resource allocation algorithms.

One key area of operating system enhancement is system security. This stems from the increasing government and private industry interest in more stringent protection measures. Numerous security-oriented facilities are currently available; these include password controls on terminals, read/write keys on files, account number restrictions, and database access control. Beyond this, additional facilities under consideration would allow the Executive to ensure a proper match between authorized users and system resources such as runs, terminals, files, and devices. These facilities include a control language for use by the security administrator in defining the security requirements of the site, and a security control module within the 1100 Executive for enforcing the security requirements. It is

anticipated that the position of the security administrator will become commonplace in the future, much as the position of the data administrator did during the mid-1970's.

In the language area, emphasis will continue to be placed on increased functionality, increased performance, and increased system integration. In part, the increased functionality stems from compliance with industry standards; e.g., it is expected that Fortran will be upgraded to comply with the latest standard expected in late 1977. Other functionality extends beyond current standards as in the case of a Codasyl structured programming specification which is under consideration for addition to the Cobol compiler. Performance enhancements will typically center around taking advantage of a newer hardware capability. Utilization of the more powerful byte instructions previously mentioned (see section on Future Architecture and Technology) would be an example. System integration encompasses the interface between the various software components discussed in this paper. Candidate enhancements include a transaction interface package send/receive interface for PL/I, and the addition of data manipulation language commands to PLUS.

In the communications area, the establishment of a distributed communications architecture (DCA) for all Sperry Univac products has not been discussed in this paper. However, new communications software is being written adhering to this architecture which is viewed as a set of development guidelines for the future. It is used to guide design in areas such as interface into other foreign networks, distribution of functions throughout the network, and utilization of intelligent terminals. Within the 1100 series (as a host), a major upgrade of the communications management system (CMS 1100) is being undertaken so that CMS properly utilizes the data and packet structures dictated by the distributed communications architecture for the interfaces throughout the network. This will include a standard front-end processor interface and the use of new software modules to support batch, demand, and transaction interfaces. Parallel development of TIP will stress performance improvements for controlling file and record placement in an enhanced transaction program environment thus making more efficient use of 1100 Executive memory management facilities.

Development activity will likewise continue in the data management system area. As databases grow in size and complexity, the need arises for additional data administrator tools to control and maintain the database. One such tool under evaluation is for the restructuring of a database. Following control language directives, it must utilize both the current schema and a "target" schema to restructure an existing database. Types of restructuring include adding or modifying areas, records and items, or sets. Another such tool under evaluation is a data dictionary which would contain additional user-supplied text describing the database. It also would include important system-generated, cross-reference information about the users of the database. For example, the Data Administrator could use the Data Dictionary to obtain a list of all programs accessing a specific data item.

One of the principal factors behind the continuing trend toward online utilization of computers is the desire to gain immediate access to a database. What often begins as a simple query, rapidly expands in complexity and scope as a user begins to gain an appreciation of the potential of direct interaction with a database. The query language processor (QLP 1100) is to be enhanced to include high-level procedure and macro facilities that, while retaining the straightforward inquiry and update facilities, offer the somewhat more advanced user a database oriented conversational programming facility.

The remote processing system (RPS 1100) is also to be expanded, particularly in terms of a screen image manipulation capability on a database. The orientation of the RPS 1100 tutorial processor, which is currently aimed toward application development by an analyst, will evolve in the direction of providing even higher-level application development facilities for the end user.

Conclusion

The 1100 Series, from the 1107 to the 1100/80 and beyond, has been traced in some detail. As the title of this paper suggests, this has been largely an evolutionary process. During the course of this evolution, we believe the 1100 series has been at the forefront of a number of technical advances. Among these are multiprogramming, symmetrical multiprocessing, virtual file control, database management, common compiler implementation techniques, and simultaneous demand, batch, and real-time operation. It is significant that these advances were made in such a way as to allow complete upward compatibility across a succession of computer models. This commitment to protect the user's software investment will continue to be a dominant factor in future extensions to the 1100 series.

Acknowledgments. Both the number of Sperry Univac people who have contributed to the development of the 1100 systems and the number of people who have made contributions to this paper are too numerous to cite. We happily acknowledge these contributions, and offer our appreciation. Contributions from the 1100 user base are both recognized and acknowledged by Sperry Univac. Special recognition is due the two major 1100 Series User Groups: USE, Inc. in the United States and UUA/E, in Europe. Also, the contributions made by Nippon Univac Kaisha are acknowledged and appreciated.

Received March 1977; revised September 1977