File No. S360-01
Order No. GA22-6907-3

IBM Systems Reference Library

IBM System/360 Model 91

Functional Characteristics

This publication describes the organization and the functional characteristics of the IBM System/360 Model 91, an information-processing system designed for ultrahigh-speed, large-scale scientific and business applications.

The system components are described, and a detailed consideration is given to the functions of processor storage, the central processing unit, the input/output channels, and the operator-control and operator-intervention portions of the system control panel. In addition, certain coding and timing considerations are discussed.

The reader is assumed to have a knowledge of information-processing systems and to have read the *IBM System/360 Principles of Operation,* Form A22-6821.

Fourth Edition (November 1971)
This is a reprint of GA22-6907-2 incorporating changes released in Technical Newsletter GN22-0300. This edition, GA22-6907-3, does not obsolete the previous edition.

Specifications contained herein are subject to change from time to time. Any such change will be reported in subsequent revisions or Technical Newsletters.

Requests for copies of IBM publications should be made to your IBM representative or to the IBM branch office serving your locality.

This manual has been prepared by the IBM Systems Development Division, Product Publications, Dept. B98, PO Box 390, Poughkeepsie, N.Y., 12602. A form is provided at the back of this publication for reader's comments. If the form has been removed, comments may be sent to the above address.

©International Business Machines Corporation 1966, 1967

Contents

The IBM System/360 Model 91 is an information-processing system designed for ultrahigh-speed, large-scale scientific and business applications. Its speed and power result primarily from the use of advanced circuit technology, the achievement of very fast access and execution times, the realization of a high degree of concurrency in operations, and the employment of highly efficient algorithms, particularly in floating-point operations.

Speed in the accessing of storage and in the execution of instructions is arrived at by the use of multiple, high-speed, interleaved main storage elements, by buffering, and by an assembly-line approach to instruction processing in which several largely autonomous execution units and a unique internal bus system play major roles.

In the Model 91, five separate units — each highly autonomous — may be operating concurrently: processor storage, storage bus control, instruction processor, fixed-point processor, and floating-point processor. Furthermore, each of these units may be performing several functions at one time. In the floating-point processor, for example, as many as three floating-point operations may be taking place concurrently.

Because of the concurrency achieved in the Model 91, the effective time required by a given instruction is not directly related to the rate at which that instruction can be processed. For example, although one normalized floating-point-add operation requires two cycles and one normalized floating-point-multiply operation requires three cycles, if the operations are logically independent, it is possible in the Model 91 to process up to two adds and one multiply concurrently for a total of three cycles rather than sequentially for a total of seven.

Although CPU operations are performed in the Model 91 in a highly parallel fashion, no special optimization is required in preparing programs for CPU processing. In general, System/360 coding will be processed in the CPU with a high degree of efficiency on the Model 91.

The Model 91 provides a major-machine-cycle time of 60 nanoseconds. Data flow is eight bytes (one double word) in parallel. The storage cycle time is 780 nanoseconds. (The cycle time of storage itself is 750 nanoseconds.) Minimum total storage access time is 600 or 900 nanoseconds, as determined by the way in which the processor storage is attached.

The logic circuitry employed in the Model 91 is ASLT (Advanced Solid Logic Technology), an outgrowth of the SLT circuitry used in other models of System/360. The advanced circuits have a basic delay time of less than 1.5 nanoseconds, as compared with SLT delay times ranging from 5 to 30 nanoseconds. In packaging, densities many times that of SLT have been achieved. Boards of approximately 8 by 12 inches can hold pluggable cards containing over 4,000 circuits. Two of these boards, for example, can contain a floating-point-add execution unit for 64 bits in which both preshifting and postshifting are accomplished.

Relationship to Other Models of the IBM System/360

Because of the emphasis on high performance, in the following cases the operation of the Model 91 differs from that specified in the *IBM System/360 Principles of Operation*, Form A22-6821.

1. The quotient of a floating-point-divide operation may differ in the Model 91 from that of other models by an amount equal to one bit in the low-order fraction position. For zero remainders, however, the results will be identical.

2. Several program interruptions that should, according to the *IBM System/360 Principles of Operation*, store a nonzero instruction-length code are imprecise in the Model 91. An imprecise interruption is one that causes an instruction-length code of zero to be stored; this code indicates that the address of the instruction causing the interruption has not been retained. When imprecise program interruptions occur, the interruption-code portion of the current PSW is used in a special way. (See the discussion of imprecise interruptions in "Instruction Processor.")

3. Because floating-point overflow and underflow cause imprecise interruptions on the Model 91, it is possible that subsequent instructions will be executed using the overflow or underflow results. For this reason, the results are made to differ from the standard System/360 results, which produce the correct fraction and a wraparound exponent. On the Model 91, overflow produces the correct sign and the maximum fraction and exponent; underflow produces a true zero result; for those instructions that change the condition code, the code will be 1 or 2 for overflow and 0 for underflow.

4. The Model 91 is capable of executing CPU stores out of sequence. Logical consistency is maintained within CPU programs, including the beginning and ending of I/O operations. However, if a CPU program is to modify a string of CCW's while they are being used by the channel, then steps must be taken to arrange the CPU program so that the stores are made in sequence.

To provide synchronization when other means are not practical, a particular branch instruction may be used which is implemented in the Model 91 in such a way that its execution is delayed until all previously decoded instructions have been completed. This branch instruction is a no-operation instruction for all models of the System/360. (See the discussion of the handling of interruptions in "Instruction Processor.")

System Components

The major components of a Model 91 are an IBM 2091 Processing Unit, an IBM 2395 Processor Storage Model 1 and 2, an IBM 2860 Selector Channel, and an IBM 2870 Multiplexer Channel. Input/output (I/O) devices are attached to the channels by means of control units. (See Figure 1.)

The four possible processor/processor-storage combinations in the Model 91 are termed K91, KK91, KL91, and L91. They differ in the number and types of processor storage units used with a 2091 Processing Unit and in the way the processor storage units are attached. (In this publication, "main storage" and "processor storage" are used interchangeably.)

PROCESSING UNIT MODEL	PROCESSOR STORAGE USED AS HIGH-PERFORMANCE MAIN STORAGE*	PROCESSOR STORAGE USED AS EXTENDED MAIN STORAGE*
2091K	One of 2395 Model 1	None
2091KK	One of 2395 Model 1	One of 2395 Model 1
2091KL	One of 2395 Model 1	One of 2395 Model 2
2091L	None	One of 2395 Model 2

* A method of attaching processor storage. See "Processor Storage."

The 2395 Model 1 is interleaved 16 ways and has a capacity of 2,097,152 bytes; the 2395 Model 2, also interleaved 16 ways, has a capacity of 4,194,304 bytes. In the identification just given of storage combinations, "K" indicates one unit of 2395 Model 1 and "L" one unit of 2395 Model 2. If the single letter "K" is used, only high-performance main storage is indicated; if the single letter "L" is used, only extended main storage is indicated. If a combination of two letters is used, both high-performance and extended main storage are indicated; the first letter designates the capacity of high-performance main storage and the second the capacity of extended main storage.

The number of bytes of storage in each of the processing unit models is:

2091K: 2,097,152
2091KK: 4,194,304
2091KL: 6,291,456
2091L: 4,194,304

An outline configuration of the 2091K Processing Unit with processor storage is shown in Figure 2. Outline configurations of the other possible combinations of processor storage units with the processor are shown in Figure 3.

The standard features for any IBM 2091 Processing Unit include:

Scientific Instruction Set (including the Standard Instruction Set and Floating-Point Arithmetic)
EDIT and EDIT AND MARK instructions. (Other decimal instructions are executed by means of interpretive program subroutines.)
Interval Timer
Protection Features (Store and Fetch Protection)
Direct Control
Display Console
2870 Multiplexer Channel Attachment
2860 Selector Channel Attachment

The display console, similar to an IBM 2250 Display Unit Model 1, is integrated with the system control panel, situated on the maintenance console (a stand-alone unit). Positioning a switch connects the display console with either an I/O channel or the system control panel. (When it is connected to a channel, the display console may be used for two-way communication. When the display console is connected to the system control panel, the communication path is from the system control panel to the display console.) For connection to an I/O channel, the display console requires one control unit position on a 2860 Selector Channel or on a selector subchannel of a 2870 Multiplexer Channel. Standard on this display console are an alphameric keyboard; a character generator; and an 8,192-byte buffer, 4,096 bytes of which are reserved for maintenance purposes and contain format control data. Additionally, the following optional features may be added to it: light pen, absolute vectors and control, programmed function keyboard, and graphic design.

An operator control panel is set into the same wall as the main maintenance console panel but is situated for use by the display console operator. To the right of this operator control panel is a power panel containing the power-on and power-off keys for the display console. (See Figures 10 and 11.)

One set of the controls and indicator lights on the

operator control panel is a standard feature. To control another System/360 processor, another set of controls and indicator lights (an optional feature) can be provided on the operator control panel.

When the resolution of the interval timer (16.7 milliseconds) is not sufficient, a high-resolution timer (an optional feature) can be used. That timer is updated by a decrementing of bit position 28 every 104 microseconds, at a frequency (9.6 kHz) that gives counting at 300 Hz in bit position 23. Because the Model 91 allows for the possibility of many operations being performed concurrently, the updating takes place with minimal interference, and no backup storage for the timer is necessary.

An attachment allows one set of the controls and indicator lights on the operator control panel to be duplicated as a remote panel on a stand-alone operator's console (IBM 2150 Console or IBM 2250 Display Unit Model 1). This attachment is a standard feature.

A channel-to-channel adapter, which is an optional feature, may be installed on an IBM 2860 Selector Channel (maximum of one per selector channel),

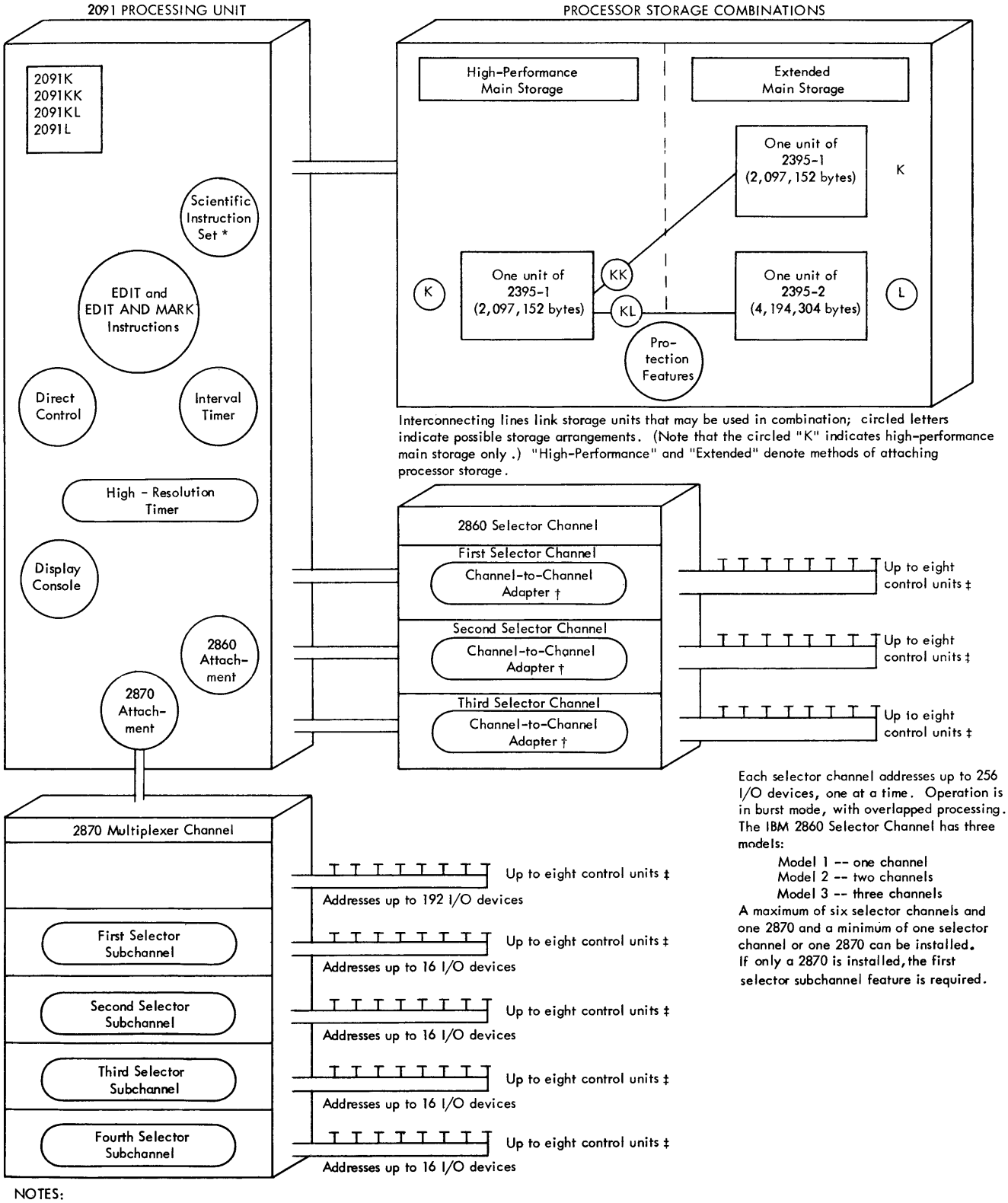
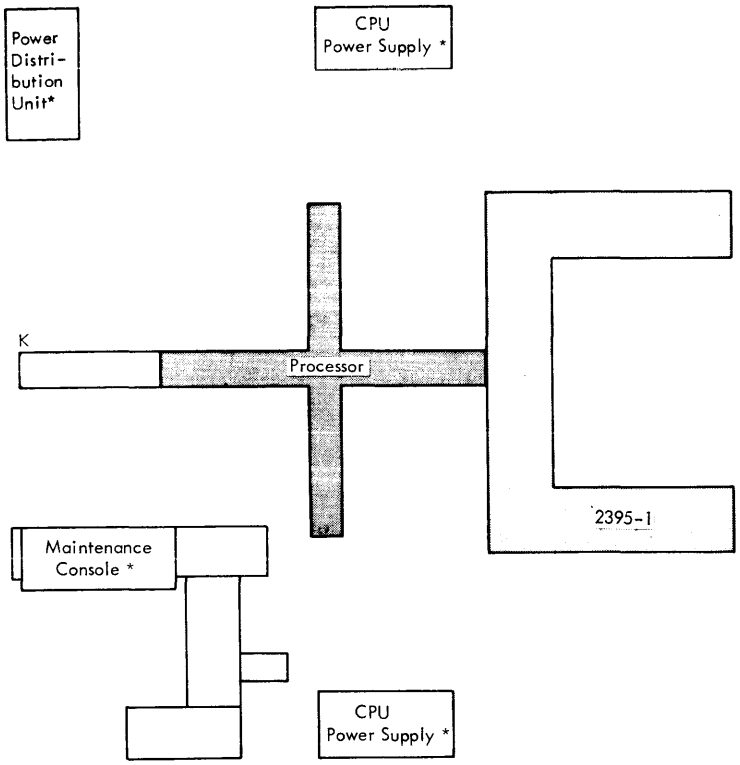


Figure 1. System/360 Model 91 Configurator



*These items may be positioned elsewhere as required. Cable limitations require that the CPU power supply units and the maintenance console be positioned no more than 15 feet from the center point of the processor, and the power distribution unit no more than 100 feet.

Figure 2. IBM 2091K Processing Unit and Processor Storage

permitting program-controlled, storage-to-storage operations to take place directly between I/O channels.

A variety of control units and input/output devices are available for use with the Model 91. Descriptions of specific input/output devices appear in separate publications. Configurators for I/O devices and for system components are also available. (See the *IBM System/360 Bibliography*, Form A22-6822.)

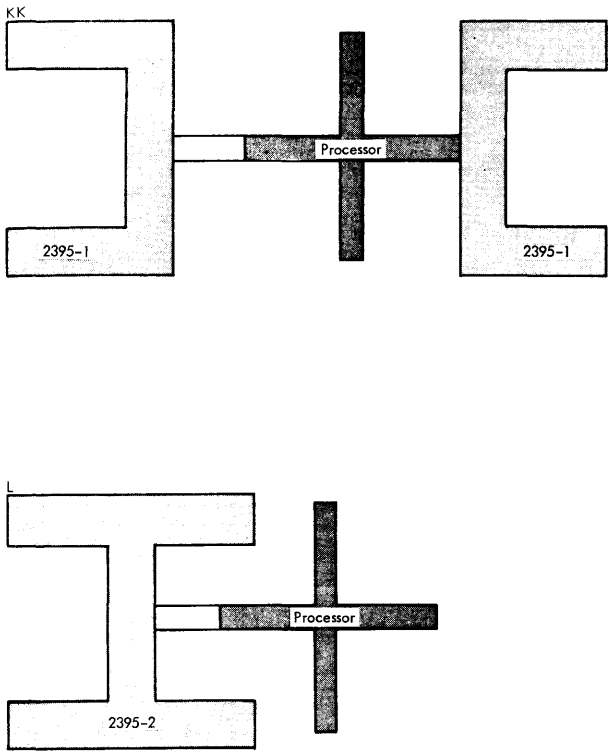
Processor Storage

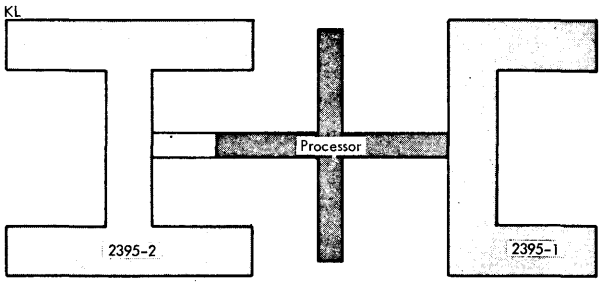
Either of two processor storage components or a combination of these can be used in a Model 91: an IBM 2395 Processor Storage Model 1 and an IBM 2395 Processor Storage Model 2. The models of processors and their related storage units are discussed under "System Components."

Processor storage may be attached in either of two ways: as high-performance main storage or as extended main storage. (In this publication, "main storage" and "processor storage" are used interchangeably.) The particular job applications determine whether the use of one or the other method of attachment or the use of both is most advantageous.

High-performance main storage offers a faster access time than that of extended main storage, in addition to the advantages of *forwarding* and *combined accessing*. Extended main storage, on the other hand, offers the potential for more storage capacity. (Forwarding is a means of reducing processor delays in retrieving information just stored. Combined accessing is an operation that eliminates the need for more than one access when more than one fetch request to the same storage location has been recognized.)

	STORAGE CYCLE TIME (NANO-SECONDS)	TOTAL ACCESS TIME (NANO-SECONDS)	INTER-LEAVING	CAPACITY (BYTES)
High-Performance Main Storage:				
One unit of 2395-1	780	600	16	2,097,152
Extended Main Storage:				
One unit of 2395-1	780	900	16	2,097,152
One unit of 2395-2	780	900	16	4,194,304





Note: Only the processor portion of the 2091 Processing Unit is shown in this figure. For outline configurations of the other elements of the 2091 (maintenance console, power distribution unit, and CPU power supply units), see Figure 2.

Figure 3. Other Processor/Processor-Storage Combinations

Storage cycle time is the length of time one of the interleaved logical units of storage will be busy whenever a reference is made to that unit. Total access time is the time required for the requested address to be sent to the logical storage unit and the information to be returned to the processor ready for use when that storage unit is not busy.

Each type of processor storage is interleaved 16 ways. In the Model 91, interleaving is the arrangement of main storage so that the addresses of successive double words are in successive, functionally independent units. (When processor storage is interleaved 16 ways, 16 successive double words are obtained from functionally separate units.) Because of this arrangement, the possibility that a reference to storage will find storage to be busy is greatly reduced, particularly when accessing is sequential.

Processing Unit

The IBM 2091 Processing Unit is the central processing unit (CPU) for the Model 91. Physically it consists of five stand-alone units: a processor, two CPU power supply units, a power distribution unit, and a maintenance console. (A motor-generator set is also provided that may be located in a remote area.) Functionally, the processing unit consists of the following major logical elements: the instruction processor, the fixed-point and variable-field-length (VFL) execution element, the floating-point execution element, the main storage control element, and the peripheral storage control element (Figure 4). The instruction processor and the two execution elements comprise the central processing element (CPE).

The instruction processor may be considered the major coordinating element in the Model 91; for each instruction it determines what must be done and which execution unit has responsibility for it. Branching, status-switching, and I/O instructions are handled by the instruction processor; other instructions are routed to other areas for further processing.

The fixed-point/VFL execution element contains the general registers, which are used also by the instruction processor. Functionally, this element operates as an independent stored-program computer; it has its own instruction stream, its own execution circuitry, and a set of operand buffers.

The floating-point execution element operates as an independent computer. Although most of the floating-point instructions require more than one cycle of execution time, this element is capable of sustaining an execution rate of up to one instruction per cycle.

The main storage control element handles all fetching and storing of data for the CPE. It is designed to minimize conflicting requests for storage and to make

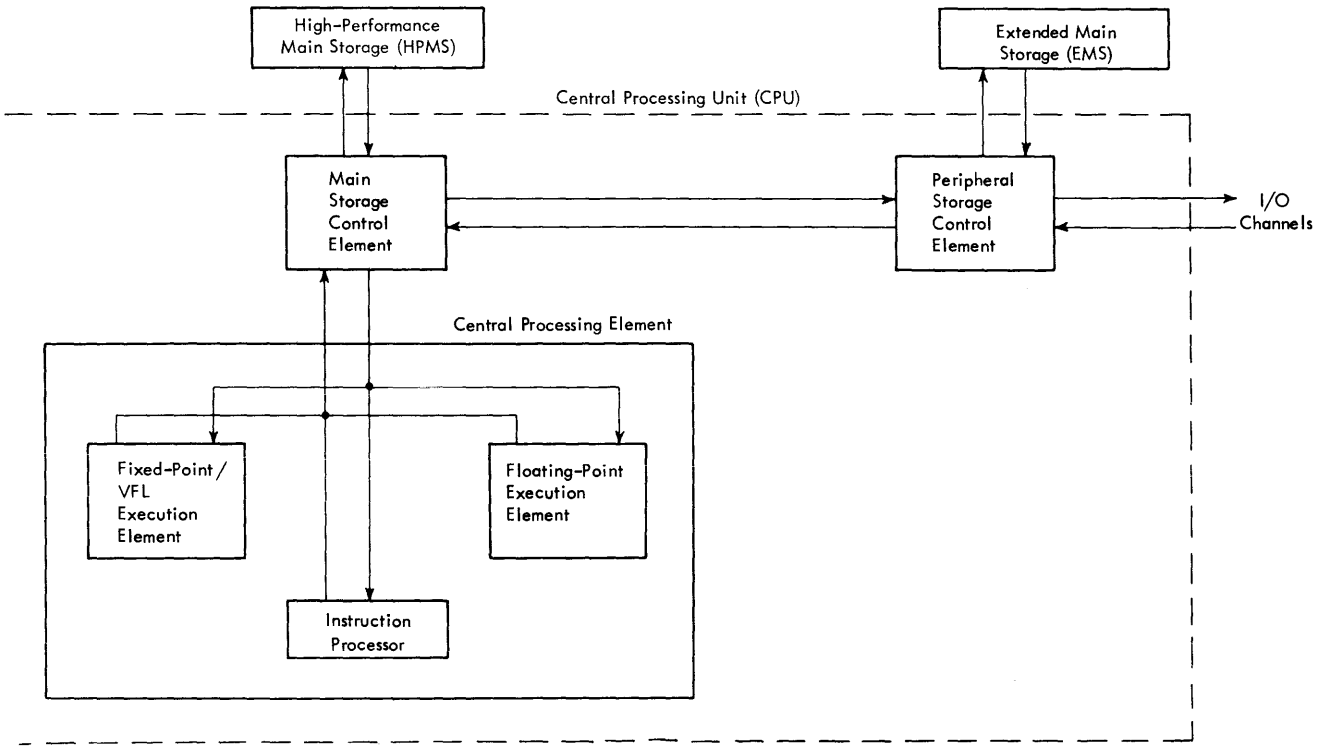


Figure 4. Model 91 Logical Elements

the most efficient use of storage. References from the CPE to extended main storage are sent to the main storage control element but are routed by it to the peripheral storage control element for handling.

The peripheral storage control element handles the transfer of information between the main storage control element, extended main storage, and input/output channels.

Instruction Processor

The primary functions of the instruction processor are the fetching and buffering of instructions from storage, the fetching of required operands, the issuance of instructions to the appropriate execution elements, the handling of interruptions, and the execution of all branching, status-switching, and I/O instructions.

The instruction processor has an instruction stack of eight double words, a set of three instruction-control registers, a set of alternate instruction buffer registers totaling two double words, a decoder, and a three-input adder for the generation of effective addresses (Figure 5). The instruction processor uses the general registers situated in the fixed-point/VFL execution element.

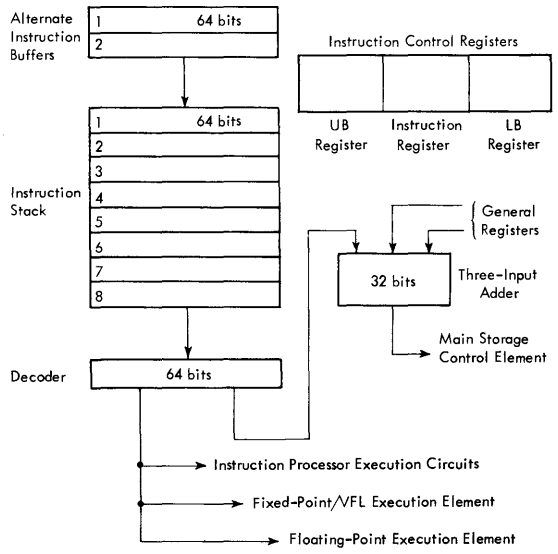


Figure 5. Instruction Processor

Instruction Fetching

Instructions fetched from storage are stored in the instruction stack of the instruction processor. An instruction stack is used for two principal reasons:

1. To minimize storage access time for instruction fetching.
2. To reduce the number of instruction fetches required while the program is executing a tight loop.

The instruction stack normally contains the current instruction double word and the next three double words; by keeping a number of double words ahead, the fetching mechanism is able to fit instruction fetches into slack periods between data fetches and stores. This mechanism minimizes the refetching of instructions when a loop backward is detected by also keeping four double words of history (instructions already decoded) in the stack.

The fetching mechanism operates differently under each of three conditions: initialization, normal operation, and recognition of a discontinuity. It is governed by three control registers: the instruction register (IR), the upper bound (UB) register, and the lower bound (LB) register. The IR points to the instruction currently being decoded, the UB register to the most recent double word brought into the stack, and the LB register to the earliest double word present in the stack.

Initialization: Initially the instruction stack is empty. When instruction fetching is initiated, the main-storage address of the first double word of instructions is set into the UB and LB registers; part of the address of the first instruction is set into the IR. The UB register, which controls the actual fetching of double words of instructions, brings the first double word into the instruction stack. At the same time the first double word is brought into the decoder.

As each instruction double word is fetched during initialization, the UB register is incremented (a double word being brought into the stack for each incrementing) until any of three conditions occurs:

1. The address in the UB register is four double words higher than that of the IR. (If the address is three higher, an additional double-word instruction fetch will be made whenever it does not delay data fetching or storing; if it is two or less, an instruction fetch will be made even if it must delay a data fetch or store.)
2. A branch instruction is decoded that sets conditional mode (see "Execution of Branching Instructions" in this section).
3. A discontinuity is recognized (see "Discontinuities" in this section).

After an instruction has been decoded, the IR is incremented by the number of halfwords of the instruction just decoded, and the next instruction is then decoded. If a double-word boundary is crossed while the IR is incremented, the UB register is incremented, causing it to fetch a new double word from storage if conditions permit fetching.

When the instruction stack is full, initialization is completed.

Normal Operation: During normal operation, the instruction fetching mechanism continually attempts to fetch double words. Fetching will not take place if any of the three conditions just described is present.

When incrementing of the UB register would cause the three low-order bits of that register to match the three low-order bits of the LB register, both the UB and LB registers are incremented. This incrementing of both registers causes the earliest double word in the stack to be replaced with the latest double word. (The three low-order bits of the LB register then point to a double word positioned one double word higher in the stack than the double word pointed to by the three low-order bits of the UB register; this positioning of pointers remains constant during normal operation.)

Discontinuities: A branch operation, interruption, or store into the instruction stream may cause a disruption in fetching. (Branching operations and interruptions are discussed separately.)

If the store instruction results in the alteration of the contents of a double word contained in the stack, the instruction fetching mechanism treats that double-word slot in the stack as empty and fetches the altered double word from storage. Because the Model 91 can be executing several instructions at one time, the instruction STORE * + 4 presents a special problem; this problem is solved in the Model 91 by making a check of the effective address of each store operation to determine whether the operation affects the next instruction following the store; if the next instruction might be affected, measures are taken to prevent it from being executed unaltered.

Instruction Issuing

In addition to fetching and buffering instructions, the instruction processor fetches the required operands and issues instructions to the appropriate execution elements.

On each machine cycle, the instruction processor checks for interlocks. If there are none, the instruction selected by the instruction register is decoded. Decoding is the first of three possible stages in the issuing of the instruction.

Stage 1: During decoding, the following is determined:

1. The type of operation to be performed.
2. Whether the operation stack for the appropriate execution element can accept the operation.
3. If a storage operand is required, whether a buffer register in the appropriate execution element is available to receive the operand; or, if a store operation is specified, whether a store address register is available in the main storage control element.
4. If an effective address is required, whether the three-input adder and general registers used in generating the effective address are available for use.

When the results of these checks indicate the instruction can be processed, the decoding control determines whether the instruction processor is operating in conditional mode (see "Execution of Branching Instructions" in this section); if it is, the operation is tagged as conditional, indicating to the execution element that it is not to decode or execute the operation until signaled to do so. The operation is then sent for processing to the appropriate execution element (usually during stage 2), along with information as to which buffer registers in the execution element, if any are needed, have been assigned by the instruction processor for use in the operation.

Stage 2: If address generation is required, the pertinent operands are routed to the three-input adder. (Another instruction can now be processed at stage 1.) If the instruction is a store, a quick check is made of the effective address and, if this check indicates a possible store into the already fetched instruction stream, processing of the instruction at stage 1 is stopped until the processor determines whether the store is actually into the instruction stream; if it is, the processing at stage 1 remains stopped until the processor has issued a fetch to storage for the instruction double word affected.

Stage 3: At this stage, the effective address of the storage operand is passed to the main storage control element. If a fetch operation is specified, the address of the buffer register to which the operand is to be sent is also specified. (During stage 3, another instruction can be processed at stage 1 and another at stage 2.)

Execution of Branching Instructions

The instruction processor executes all branching instructions. The actions taken by the instruction processor as a result of decoding a branch instruction are determined by the type of branch instruction to be processed, the availability of circuitry for processing, and the following:

1. Whether the instruction processor is in conditional mode (see "Conditional Mode" in this section).
2. Whether the instruction processor is in loop mode (see "Loop Mode" in this section).
3. If loop mode is established, whether the current instruction is that which defined the current loop.
4. Whether the current instruction is the target of an EXECUTE instruction currently being processed.

Normally when a branch is taken, the target address of the branch is set into the instruction register, and the UB and LB registers and instruction stack are adjusted as required.

When a conditional branch is encountered and loop mode is not set, the instruction processor operates as

though either direction could be taken but assumes that the strongest possibility is that the branch will not be taken. It continues to process instructions in the instruction stack as long as conditions permit, while issuing operations to the fixed-point and floating-point execution elements on a conditional basis; these conditional operations will not be executed until after the condition code is set. Also, the instruction processor makes use of two alternate instruction registers. Into these registers it fetches the branch-target double word and the succeeding double word. Therefore, regardless of the outcome of the branch operation, the instruction processor will have a lead in the correct direction.

Conditional Mode: Conditional mode is established when the instruction processor executes a BRANCH ON CONDITION instruction for which the condition code is not yet determined.

When conditional mode is set, no additional instruction fetches are made. The instruction processor continues to decode instructions, generates addresses, and issues operations to the fixed-point and floating-point execution elements; the operations issued, however, are tagged as conditional and cannot be decoded or executed until the condition code is set and the instruction processor sends a signal to the execution element.

The instruction processor continues to decode instructions conditionally until any of the following conditions occurs:

1. The condition code is set.
2. No instructions are available in the instruction stack.
3. The operation stack of the fixed-point or floating-point execution element is full, and the currently decoded instruction requires that filled execution element.
4. An instruction to be executed within the instruction processor is decoded, or a variable-field-length instruction is decoded. (However, a no-operation instruction or an unconditional branch may be executed during conditional mode.)

Loop Mode: Whenever a branch backward is taken to a target fewer than eight double words back from the current address in the instruction register, loop mode is entered. When loop mode is entered, the loop is locked into the instruction stack and, as a result, can be executed repetitively without refetching of the instructions. Thus, conflicts between instruction fetching and data fetching are eliminated, and branches can be executed faster.

During loop mode, if there are no data fetches or stores to be made, an instruction double word is fetched every cycle until the instruction stack is full. If data fetches or stores are to be made, instruction double-word fetches are made every other cycle.

When loop mode is entered, the branch target address is placed in one special register and the address of the branch instruction in another special register. When another branch instruction is decoded during loop mode, the current instruction address is compared with the address (in the special register) of the branch instruction that initiated loop mode; if they are the same, the branch is made to the target address in the other special register. Because no time has been taken to re-form the address specified in the branch instruction, one cycle has been saved.

If a conditional branch instruction is processed when loop mode is already set, it is assumed that the branch will be taken; therefore, during loop mode no temporary fetches (down the no-branch path) are made for conditional branches.

Loop mode is turned off when any of the following conditions occurs:

1. A branch out of the instruction stack is taken.
2. The instruction processor starts to decode the 32nd halfword in the instruction stack.
3. The target of the quick loop is the same as the target of the outermost loop, and the branch closing the quick loop is not taken. (If a set of nested loops fits in the instruction stack, the innermost loop is called the quick loop.)
4. The base register or index register of the quick-loop branch is altered.

Programming Notes: Because of item 2, a loop with 29-31 halfwords may be aligned on a double-word boundary by the use of a conditional-no-operation (CNOP) instruction. If the loop has fewer than 29 halfwords, it is executed in loop mode; if it has more than 31 halfwords, it is not executed in loop mode.

Because of item 3, if the nested loops of a set all have the same target address, loop mode will be destroyed every time an exit is made from the quick loop. To prevent loop mode from being destroyed, a no-operation instruction may be used as a dummy branch point for one of the outer loops.

Execution of Other Instructions

The instruction processor executes all status-switching and I/O instructions and plays a large part in the execution of multiple-operation instructions.

When one of these instructions is processed, the instruction processor usually does not issue any succeeding instruction until its part in processing the first instruction is completed.

None of these instructions is executed while conditional mode is set, and many of the instructions require that all instructions being executed when that

instruction is decoded be completed prior to its execution. The instructions requiring this completion of other instructions are the four I/O instructions and LOAD PSW, SUPERVISOR CALL, SET STORAGE KEY, and SET PROGRAM MASK (except when the old and new mask bits are the same). (Also, one type of BRANCH ON CONDITION instruction, a no-operation instruction, is implemented in the Model 91 in such a way that all other instructions being executed when it is decoded must be completed prior to its execution. See the programming note in "Handling of Interruptions" in this section.)

The following instructions are classed as multiple-operation instructions:

Load Multiple (LM)
Store Multiple (STM)
Translate (TR)
Translate and Test (TRT)
And (NC)
Or (OC)
Exclusive Or (XC)
Compare Logical (CLC)
Move Zones (MVZ)
Move Numerics (MVN)
Move (MVC)
Move With Offset (MVO)
Pack (PACK)
Unpack (UNPK)
Edit (ED)
Edit and Mark (EDMK)

These multiple-operation instructions have variable length data fields and require the issuing of more than one operation from the instruction processor to the fixed-point execution element, with which the instruction processor shares responsibility for execution. Also, each operation of a multiple-operation instruction issued to the fixed-point area contains information concerning at least one storage request.

The multiple-operation instructions are the only instructions except CONVERT TO BINARY that cause operands to be fetched into the floating-point operand buffers for use in the fixed-point area. Furthermore, four of the six fixed-point operand buffers are unavailable for reassignment while a multiple-operation instruction is being executed.

Usually the instruction processor is available to issue the succeeding instruction after it has issued the last required operation to the fixed-point area. If the next instruction is in the SI format, it is not issued until the variable-field-length execution for the multiple-operation instruction is complete. If the multiple-operation instruction is a TRANSLATE AND TEST (TRT) or an EDIT AND MARK (EDMK) instruction, the instruction processor will be available to issue subsequent instructions only after the entire TRT or EDMK instruction has been executed.

Handling of Interruptions

The Model 91 performs all the interruption functions defined for the IBM System/360. (See the *IBM System/360 Principles of Operation,* Form A22-6821.) The supervisor call, external, machine check, and I/O interruptions are logically handled as defined. However, the performance objectives of the Model 91 require some deviations in the handling of program exceptions. The program-exception deviations are basically those resulting from an operation that has been sent by the instruction processor to another element for execution, and the current PSW no longer references the operation. Consequently, the interruption-causing instruction cannot be directly identified, and such a program interruption is described as being imprecise. An imprecise interruption is identified by the storing of zero as the instruction-length code in the PSW current at the time of interruption.

The instruction-length codes (ILC) for program interruptions on the Model 91 follow. The codes in this listing should be considered to replace, for the Model 91, those listed for ILC on program interruptions in the *IBM System/360 Principles of Operation.*

PROGRAM EXCEPTION	ILC
Operation	1, 2, 3
Privileged Operation	1, 2
Execute	2
Protection	0
Addressing	0, 1, 2, 3
Specification	0, 1, 2, 3
Data	0
Fixed-Point Overflow	0
Fixed-Point Divide	0
Decimal Divide	Not applicable
Decimal Overflow	Not applicable
Exponent Overflow	0
Exponent Underflow	0
Significance	0
Floating-Point Divide	0

Note that in this listing the following program exceptions always cause *imprecise* interruptions:

1. Data, fixed-point-overflow, and fixed-point-divide exceptions signaled from the fixed-point/VFL execution element.

2. Exponent-overflow, exponent-underflow, significance, and floating-point-divide exceptions signaled from the floating-point execution element.

3. A protection exception signaled from either the main storage control element or the peripheral storage control element when a protection violation is detected.

Note also that two exceptions — addressing and specification — can produce either precise or imprecise program interruptions, as determined by the problem.

An addressing exception resulting in a *precise* program interruption is produced if any of the following conditions is detected:

1. Any portion of the current instruction to be decoded lies outside available storage.
2. The address generated for any of the following instructions lies outside available storage: READ DIRECT, WRITE DIRECT, LOAD PSW, SET SYSTEM MASK, SET STORAGE KEY, INSERT STORAGE KEY, and DIAGNOSE.
3. Any portion of the target instruction for EXECUTE lies outside available storage.

All other addressing exceptions, which are signaled after the completion of address generation leading to the fetching or storing outside of available storage, result in *imprecise* program interruptions.

A specification exception resulting in a *precise* program interruption is produced if any of the following conditions is detected:

1. An attempt is made to execute an instruction specified at an odd-numbered location in storage.
2. The R_1 field of an instruction specifies an odd-numbered register for the pair of general registers that contains a 64-bit operand.
3. A number other than 0, 2, 4, or 6 is specified for a floating-point register.
4. The block address specified in SET STORAGE KEY or INSERT STORAGE KEY has the four low-order bits not all zero.
5. The three low-order bits are not all zero in the address generated for LOAD PSW or DIAGNOSE.

All other specification exceptions, which are signaled after the completion of address generation and arise from improper boundary resolution for data fetching and storing, result in *imprecise* program interruptions. Because of the imprecise interruption, the specification exceptions that arise from improper boundary resolution cause termination of the operation; therefore, the data in registers and main storage may have been altered.

When an imprecise interruption is signaled, the instruction processor ensures that all instructions that were decoded before the signal was recognized are completed before the interruption is honored. When the interruption is taken, the instruction address stored in the program old PSW points to the next instruction which would have been decoded, and for which an attempt would have been made to issue it, had the interruption not occurred.

When the program interruption is precise, bits 28-31 of the program old PSW identify the exception causing the interruption, the remainder of the interruption code (bits 16-27) is all zeros, and the instruction-length code (bits 32-33) is 1, 2, or 3, as appropriate.

When an imprecise interruption takes place, not just one but several exceptions may have occurred because all decoded instructions are completed before the interruption is taken. Also, because instructions can be executed out of sequence, the interruption condition recognized first may not be the condition that logically should be recognized first. To account for both possibilities (an out-of-sequence detection and the occurrence of more than one type of exception, either within one or different instructions), the following action is taken when an imprecise interruption occurs: each type of exception that took place is identified in bits 16-25 of the program old PSW, and bits 26-31 are set to zero. Also, the instruction-length code (bits 32-33) is set to zero.

BIT POSITION IN PROGRAM OLD PSW	PROGRAM EXCEPTION
16	Protection
17	Addressing
18	Specification
19	Data
20	Fixed-Point Overflow
21	Fixed-Point Divide
22	Exponent Overflow
23	Exponent Underflow
24	Significance
25	Floating-Point Divide

NOTE: For an imprecise interruption, the types of exceptions that occurred, but not the number of exceptions of any one type that occurred, are identified in the program old PSW.

A logical consistency is maintained when a precise program interruption precedes an imprecise program interruption that logically should have taken place first. Should an imprecise interruption occur during the execution of outstanding instructions prior to honoring a precisely identifiable interruption, the instruction causing the precise interruption is not executed, the precise interruption condition associated with this instruction is not indicated, and the address of the instruction causing the precise interruption is placed in the instruction-address portion of the program old PSW. In effect, the instruction leading to the precise interruption is treated as never having occurred, and a return to the program causes the original interrupting instruction to be reinitiated. (The same operation takes place when a supervisor-call interruption is followed by an imprecise program interruption that logically should have occurred first.)

Imprecise interruptions that arise from conditional instructions (that is, instructions issued subsequent to a BRANCH ON CONDITION instruction for which the condition code is not yet determined) are noted and either activated or canceled, as appropriate, when the conditional instructions themselves are activated or canceled.

Logical accuracy is preserved in all situations where a basic machine status change is involved. For exam-

ple, all instructions issued under a program mask are completed prior to changing the mask, ensuring that the mask stored is that which allowed the interruption.

Programming Notes: A program may not operate correctly on the Model 91 if it requires identification of the instruction causing an imprecise interruption. When an imprecise interruption occurs, the program old PSW does not reference the operation that caused it.

A program also may not operate correctly on the Model 91 if it requires the honoring of an imprecise interruption before some instruction later in the program is executed. When an imprecise interruption is detected, all instructions decoded by that time are executed before the interruption is taken. Therefore, several instructions following the instruction that caused the imprecise interruption may be executed before the interruption is taken. (How many of these subsequent instructions will be executed will vary from instance to instance, largely because the Model 91 can execute instructions concurrently and out of sequence.) It is possible, at the programmer's option, to return to the problem program, but because all decoded instructions are completed before the interruption is taken, the instructions executed subsequent to the interruption may have been adversely affected by the program exception.

If preciseness is a principal concern, the unwanted effects of imprecise program interruptions can usually be eliminated by testing and masking, as appropriate, and by using this BRANCH ON CONDITION instruction:

MNEMONIC	TYPE	M1 FIELD	R2 FIELD
BCR	RR	Not zero	Zero

This branch instruction is a no-operation instruction for all of System/360 but is implemented in the Model 91 in such a way that its execution is delayed until all previously decoded instructions have been completed.

NOTE: The use of this no-operation instruction degrades the performance of the Model 91; it should be used only to eliminate a problem for which there is no other reasonable solution.

Note that a program may naturally have been arranged so that the adverse effects of certain imprecise program interruptions are eliminated in advance. For example, in addition to the branch (no-operation) instruction just mentioned, the execution of the following instructions is delayed until all previously decoded instructions have been completed: the four I/O instructions, LOAD PSW, SUPERVISOR CALL, SET STORAGE KEY, DIAGNOSE, and SET PROGRAM MASK (except when the old and new mask bits are the same).

There is a situation other than the one concerning imprecise interruptions in which the execution of instructions out of sequence may present a problem. Although the CPU maintains a logical consistency with respect to its own operations, including the starting and ending of I/O operations, it cannot ensure logical consistency for asynchronous units during their operations. For example, if an I/O channel program relies on the proper sequencing of stores by the CPU in ensuring proper channel operation, then steps must be taken in the CPU program to guarantee that the stores actually are made in sequence. The no-operation instruction can be used to accomplish this.

Main Storage Control Element

The main storage control element (MSCE) controls the accessing of high-performance main storage (HPMS) by the central processing element (CPE) and the peripheral storage control element (PSCE). Also, it provides a link to the PSCE for accesses to extended main storage by the CPE. (See Figure 4.)

The principal functions performed by the MSCE are:

1. The buffering of fetches and stores to busy storage.
2. The appropriate sequencing of stores and fetches to the same address.
3. *Forwarding* when a fetch follows a store to the same address.
4. *Combined accessing* when more than one fetch request is to the same address.
5. The routing of data from storage to buffers of the CPE and the PSCE.

The MSCE consists of the following major elements (Figure 6):

1. Three store address registers (SAR).
2. Three corresponding store data buffers (SDB).
3. A request stack of four positions.
4. An accept stack that contains up to five addresses but allows up to 13 storage modules to operate concurrently.
5. A storage address bus.

CPE Storage Requests

Fetching: A CPE fetch request is made either to high-performance main storage (HPMS) or to extended main storage via the PSCE. A fetch request from the CPE is issued only if the MSCE can retain the request if it is rejected. Normally the request is accepted immediately. If it is to HPMS, it is sent directly to the storage module and to the MSCE accept stack; if it is to extended main storage, it is sent to the PSCE.

The MSCE accept stack is capable of controlling 13 concurrently operating storage modules of HPMS.

By the use of control information supplied with the fetch request, the fetched information is directed to the instruction or operand buffer in the CPE that was reserved in advance for the particular fetch.

If the HPMS module or the PSCE is busy, the CPE fetch request is rejected and sent to the MSCE request

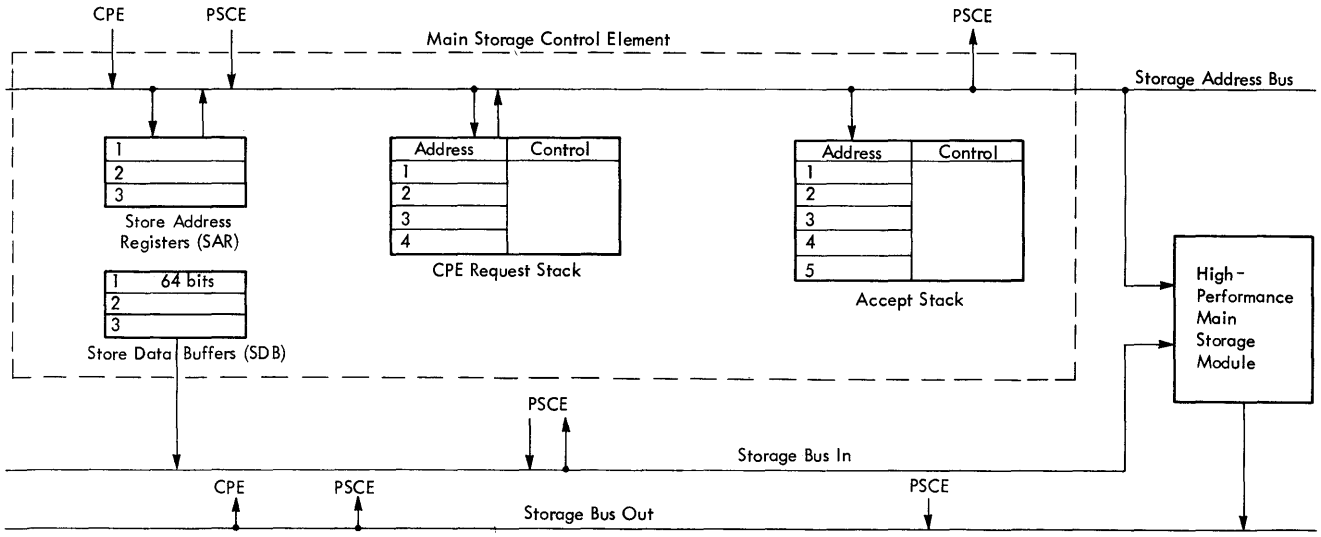


Figure 6. Main Storage Control Element

stack, which can retain four rejected requests. When the HPMS module or the PSCE is no longer busy, the first-in request in the stack is accepted. If HPMS is addressed, the request goes to both the HPMS module and the MSCE accept stack.

Storing: A CPE store request is issued only if a store address register (SAR) is available. When the CPE store request is initiated, the store address is placed in one of the three SAR's and the data to be stored are placed in the corresponding store data buffer (SDB) when the data become available. When the data are in the SDB and when a position in the request stack is available (to retain the request if it is rejected), the store request is re-initiated. If an HPMS module is addressed and is not busy, the store address is placed in the accept stack and sent to storage, and the data are sent from the SDB to storage. If the PSCE is addressed and is not busy, the store address is sent from the SAR to the PSCE, and the data are sent from the SDB to the PSCE.

If the HPMS module (or the PSCE) is busy, the store request is placed in the MSCE request stack. Then, when the HPMS module (or the PSCE) is no longer busy, the first-in request in the stack is accepted, and the operations described earlier are initiated.

PSCE Storage Requests

A fetch or store request from the PSCE to the HPMS is issued only when the storage module addressed is not busy. Therefore, each PSCE request has guaranteed acceptance. When it is accepted, the PSCE request is placed in the MSCE accept stack in addition to being sent to the HPMS module.

Multiple CPE Requests to Storage

The storage address in each new CPE fetch request to HPMS that is presented to the MSCE is compared to the three addresses in the SAR, the four in the request stack, and the five in the accept stack.

If the address matches one in the SAR, the request is rejected, placed in the request stack, and tagged for *forwarding*, which consists in (1) linking the fetch to the store in such a way that the fetch logically follows the store and (2) placing the fetch destination address with the store address in the accept stack so that the information is sent from the HPMS module to the CPE when the store is completed. As a result, part of the access time normally required for the fetch is eliminated.

If the address matches one in the accept stack, it is also placed in the accept stack and tagged for *combined accessing*, which will result in the satisfying of more than one fetch request to the same address by only one selection of the storage module.

If the address matches one in the request stack, it is placed in the request stack and tagged for a future combined-accessing operation.

Fixed-Point and Variable-Field-Length Execution Element

The fixed-point and variable-field-length (VFL) execution element executes all fixed-point arithmetic, logical, and variable-field-length arithmetic operations. It consists of five major logical elements (Figure 7):

1. An operation stack (FXOS) of six positions
2. Sixteen general registers
3. Six 32-bit operand buffers (FXB)
4. A fixed-point execution unit
5. A VFL execution unit

Fetches for data fields from storage that are necessary for the processing of a fixed-point operation are initiated by the instruction processor, which also reserves the buffers in the fixed-point area that are to receive the operands requested. The instruction proc-

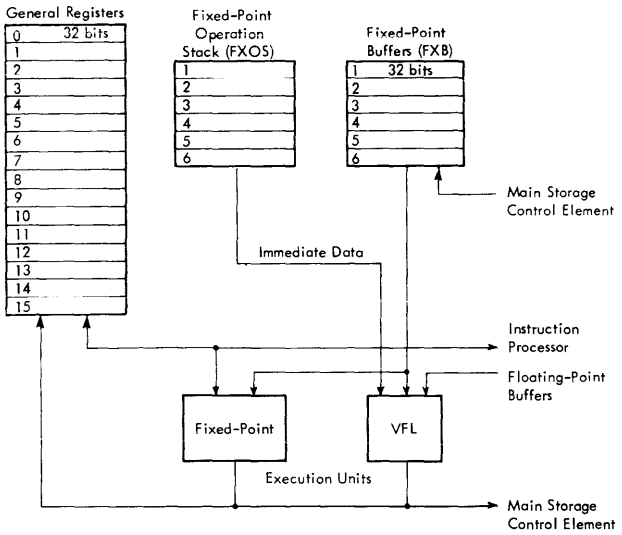


Figure 7. Fixed-Point/VFL Execution Element

essor also maintains counters for its own use that indicate whether the fixed-point operation stack has an available position, which fixed-point buffers are available, and which general registers are available to the instructor processor and which are being used by the fixed-point/VFL execution element.

During normal processing, operations in the FXOS are decoded serially and issued to either the fixed-point execution unit or the VFL execution unit. An operation can be executed if it has been decoded, if the data are available, and if the execution circuitry is free. When decoding is completed, the instruction processor is notified that the stack position and operand buffers assigned to that operation are free.

The execution of one operation is overlapped when possible with the decoding of the next. When a multiple-operation instruction is processed (see the discussion of the execution of multiple-operation instructions in "Instruction Processor"), decoding of the next instruction in the FXOS does not begin until the execution of the last of the multiple operations is begun.

Operations tagged as conditional are not decoded or executed until they are activated or canceled by the instruction processor. A canceled operation is decoded in one cycle, and execution of the operation consists in the freeing of any operand buffers previously assigned to the canceled operation without actual execution of the operation.

At any time during a fixed-point/VFL operation, the instruction processor can request a direct store into the general registers, an operation which, because the instruction processor has priority, may delay fixed-point/VFL processing.

Fetches made during the execution of a multiple-operation instruction may require the use of operand buffers in the floating-point execution element; also, four of the six fixed-point operand buffers are unavailable for reassignment while such an instruction is being processed.

Floating-Point Execution Element

The floating-point execution element handles the execution of all floating-point arithmetic operations. It can execute several operations at one time (for a maximum of two adds and one multiply or divide) if the operations are logically independent; this performance capability results largely from three significant features: the use of (1) operand and instruction buffering, (2) multiple execution units employing extremely efficient algorithms, and (3) a common data bus, which links the several sets of execution circuitry in such a way that the full power of the multiple execution units is realized without a reliance on programming for the special arrangement of instructions.

The floating-point area contains the following major logical elements (Figure 8):

1. An operation stack (FLOS) of eight positions.
2. Four floating-point registers (FLR).
3. Six operand buffers (FLB), which are also used by the fixed-point area when any multiple-operation instruction or the CONVERT TO BINARY instruction is processed.
4. Two execution units: one add unit (preceded by three reservation stations) capable of performing two add operations concurrently and one multiply/divide (M/D) unit (preceded by two reservation stations).

Decoding of operations in the FLOS proceeds serially. As an execution unit is selected for an operation (on the second cycle), the decoding of the next operation (on the first cycle) can begin. The FLOS issues operations subject to only one principal constraint: a reservation station of the appropriate type must be available.

The FLOS need not wait for all the operands to be available (as in the fixed-point area) before issuing the operation. Instead, the selected reservation station and controls hold the issued operation until the required operands have been collected and then engage the execution circuitry.

Because several operations may be in various stages of execution at one time, provisions must be made for the proper sequencing of dependent operations. A system of tagging for usage of the common bus is used to ensure the proper sequencing of these operations. Also, the system facilitates the out-of-sequence execution of independent operations.

The FLB in conjunction with the common data bus execute all RX load operations. RR load and RR load and test operations are executed by the common data bus

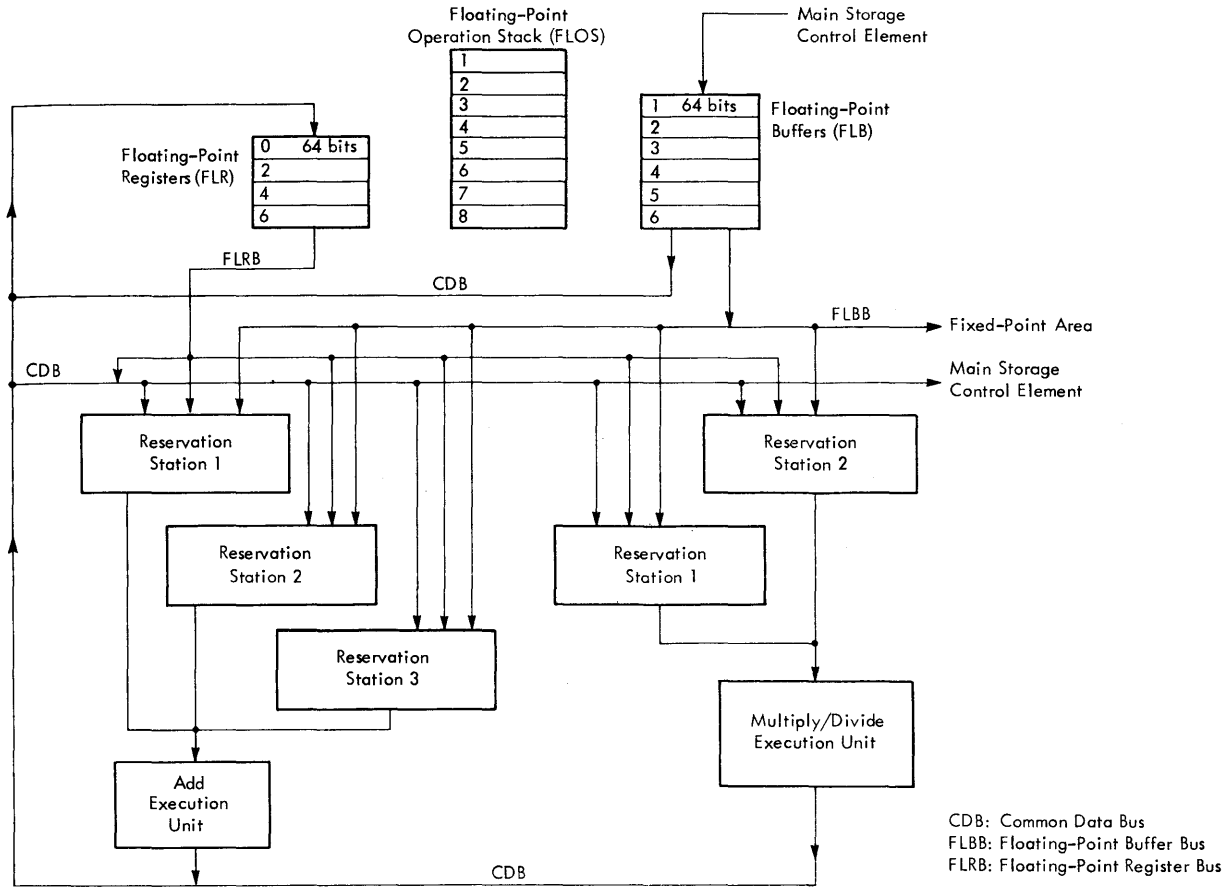


Figure 8. Floating-Point Execution Element

in conjunction with special testing circuitry. Store operations are executed by the three store data buffers. Multiply and divide operations are executed by an M/D unit. All other operations are executed by the add unit.

An add execution unit can begin if the operation has been decoded, the data are available, and another add operation of higher priority is not beginning on the same cycle. Two add operations can be executed concurrently by offsetting the start of the second operation one cycle from the start of the first. While two operations are being performed, the third reservation station may be acquiring data.

A multiply or divide execution can begin if the operation has been decoded, the data are available, another multiply or divide operation of higher priority is not beginning on the same cycle, and the execution circuitry is free. The two M/D reservation stations share a single execution section; therefore, only one M/D operation may be executed at one time.

Fetches for data fields that are necessary for the processing of a floating-point operation are initiated by the instruction processor, which also reserves the buffers in the floating-point area that are to receive the operands requested. The instruction processor also maintains counters for its own use that indicate whether the FLOS has an available position and which floating-point buffers are available. When the FLOS completes decoding, it signals the instruction processor that the stack position is empty. If an operation has been decoded, the related operand buffer is set free at the time it is filled; if the operand buffer is filled before the related operation is decoded, however, the buffer is set free one cycle after completion of the decoding.

Operations tagged as conditional are not decoded or executed until they are activated or canceled by the instruction processor. A canceled operation is decoded in one cycle, and execution of the operation consists in the freeing of all operand buffers previously assigned to the canceled operation without actual execution of the operation.

Peripheral Storage Control Element

The peripheral storage control element (PSCE) controls the transfer of information between the main storage control element, extended main storage, and input/output channels.

The main storage control element (MSCE) sends to the PSCE all requests to extended main storage that originate in the central processing element. In a like manner, the MSCE services all requests from the PSCE to high-performance main storage.

The PSCE is logically composed of the bus control and the common channel control.

Bus Control

All accesses to extended main storage, as well as the accesses to high-performance main storage from I/O channels or the storage channel, are controlled and sequenced by the bus control.

To make maximum and efficient use of extended main storage, the bus control maintains a request stack consisting of eight buffer registers. As a module of extended main storage becomes available, the bus control scans the request stack for its next request and, finding one, initiates a storage cycle.

Common Channel Control

The common channel control provides for the attachment of up to six selector channels and one multiplexer channel (see "Channels"). It monitors all channel storage requests and controls the sequencing of I/O instructions and interruptions. In addition, when high-speed I/O devices fetch from main storage, it initiates and controls prefetching in order to minimize the possibility of overrun.

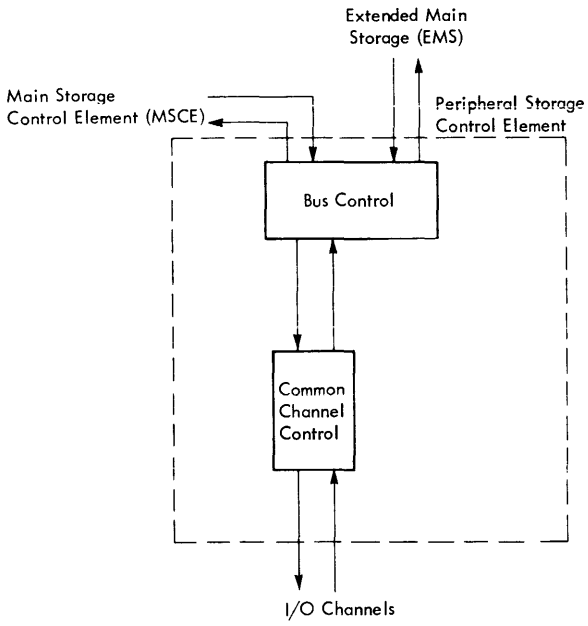


Figure 9. Peripheral Storage Control Element

Channels

In the Model 91 the IBM 2860 Selector Channel and the IBM 2870 Multiplexer Channel are available for use. The selector and multiplexer channels provide for the attachment of I/O devices to the system. The channel relieves the CPU of the task of communicating directly with the I/O devices and permits data processing to proceed concurrently with I/O operations.

Data are transferred a byte at a time between the I/O device and the channel. A standard channel-to-control-unit interface provides a uniform method of attaching control units to all channels. Data transfers between the channel and the PSCE are eight bytes (one double word) in parallel for both selector and multiplexer channels.

2860 Selector Channel

The 2860 Selector Channel provides for the attachment and control of I/O control units and associated devices. It is available in three models:

Model 1 – provides one selector channel
Model 2 – provides two selector channels
Model 3 – provides three selector channels

In addition to one 2870 Multiplexer Channel, a total of six selector channels in the following combinations may be attached:

NO. OF CHANNELS REQUIRED	RECOMMENDED COMBINATIONS
1	One of 2860-1
2	One of 2860-2
3	One of 2860-3
4	One of 2860-3 and one of 2860-1; or two of 2860-2
5	One of 2860-3 and one of 2860-2
6	Two of 2860-3

At least one 2860 (any model) is required if no 2870 Mutiplexer Channel is attached. If only a 2870 is attached, the first selector subchannel feature is required.

The selector channel permits data rates of up to 1.3 million bytes a second. I/O operations are overlapped with processing, and depending on the data rates and channel programming considerations, all selector channels can operate concurrently. A set of channel control and buffer registers permits each channel to operate with minimal interference.

A maximum of eight control units can be attached to each selector channel. Each control unit may have more than one I/O device connected to it, but only one device per channel may transfer data at any given time. A selector channel operates only in burst mode.

Channel-to-Channel Adapter Feature

A channel-to-channel adapter is available as an optional feature. The adapter provides a path for operations to take place between two channels and synchronizes those operations. It may be used in multiple-processor or single-processor systems: in a multisystem to achieve rapid communications between the channels of two System/360 models and in a single system to move blocks of data from one area in main storage to another area in main storage.

The adapter uses one control-unit position on each of the two channels. Only one of the two connected channels requires the feature, and in the Model 91 one adapter may be installed per selector channel.

For restrictions on channel attachments for another IBM System/360 model used with the Model 91, refer to the Systems Reference Library (SRL) functional characteristics publication for that particular System/360 model.

2870 Multiplexer Channel

The 2870 Multiplexer Channel provides for the attachment of a wide range of low-speed to medium-speed I/O control units and associated devices. One 2870 can be attached to the Model 91.

The multiplexer channel provides up to 196 subchannels, including four selector subchannels. The basic multiplexer channel has 192 subchannels; it can attach eight control units and can address 192 I/O devices. The basic mutiplexer channel can operate several multiplex-mode I/O devices concurrently, or a single burst-mode device may be operated.

One to four selector subchannels are optional with a 2870. Each selector subchannel can operate one I/O device concurrently with the basic multiplexer channel. Each selector subchannel permits attachment of eight control units for certain devices having a data rate not exceeding 180 kilobytes (kb) a second. Regardless of the number of control units attached, a maximum of 16 I/O devices can be attached to a selector subchannel.

The maximum aggregate data rate for the multiplexer channel ranges from 110 kb to 670 kb, depending on the number of selector subchannels in operation. The first three selector subchannels may operate concurrently at up to 180 kb for each subchannel; when all four selector subchannels operate concurrently, the fourth has a maximum data rate of 100 kb. Each selector subchannel in operation diminishes the basic multiplexer channel's maximum data rate of 110 kb; the maximum data rates for concurrent selector subchannel operations are:

BASIC MULTIPLEXER CHANNEL	DATA RATES FOR SELECTOR SUBCHANNELS 1st	2nd	3rd	4th
110 kb				
88 kb	180 kb			
66 kb	180 kb	180 kb		
44 kb	180 kb	180 kb	180 kb	
30 kb	180 kb	180 kb	180 kb	100 kb

NOTE: The 180-kb maximum data rate for 2870 selector subchannels pertains to attachment of magnetic tape devices; timing factors other than data rates may preclude attachment of direct-access storage devices having lesser data rates. Also note that when other channels in addition to the 2870 are in operation, the total system I/O data rate must be analyzed.

Channel-to-Channel Adapter Connection to 2870

The 2870 may be connected to another system channel. The channel-to-channel adapter, however, is installed on the other channel, not on the 2870.

System Control Panel

The system control panel, situated on the maintenance console, contains the switches, keys, and indicator lights necessary to operate, display, and control the system. The system consists of the CPU, storage, channels, on-line control units, and input/output devices. Off-line control units and I/O devices, although a part of the system environment, are not considered part of the system proper.

System controls are logically divided into three classes: operator control, operator intervention, and customer engineering control. This section of the manual discusses operator control and operator intervention.

By the use of the control panel, the operator can perform the following system control functions:

1. Reset the system.
2. Store and display information in storage, registers, and program status word (PSW).
3. Load initial program information.

System Control Functions

System Reset

The system-reset function resets the CPU, channels, and on-line nonshared control units and I/O devices.

The CPU is placed in the stopped state, and all pending interruptions are eliminated. All error-status indicators are reset to zero.

In general, the system is placed in such a state that processing can be initiated without the occurrence of machine checks, except those caused by subsequent machine malfunction.

The reset state for a control unit or device is described in the appropriate Systems Reference Library (SRL) publication. A system-reset signal from a CPU resets only the functions in a shared control unit or device belonging to that CPU. Any function pertaining to another CPU remains undisturbed.

The system-reset function is performed when the system-reset key is pressed, when the PSW-restart key is pressed, when initial program loading is initiated, or when a power-on sequence is performed.

Programming Note: If a system reset occurs in the middle of an operation, the contents of the PSW and of the result registers or storage locations are unpredictable. If the CPU is in the wait state when the system reset is performed, and no I/O operation is in progress, this uncertainty is eliminated.

A system reset does not correct parity in registers or storage. Because a machine check occurs when information with incorrect parity is used, the incorrect information should be replaced by loading new information.

Store and Display

The store-and-display function permits manual intervention in the progress of a program. The storing and/or displaying of data may be provided by a supervisor program in conjunction with proper I/O equipment and the interrupt key.

In the absence of an appropriate supervisor program, the controls on the operator intervention panels allow direct storing and displaying of data. This is done by placing the CPU in the stopped state and subsequently storing and/or displaying information in main storage, in general and floating-point registers, and in the instruction-address part of the PSW. The stopped state is achieved when the stop key is pressed, when single instruction execution is specified and the instruction has been executed, or when a preset address is reached. In the Model 91 the transition from operating to stopped state includes completing all instructions that were decoded at the time stopped state was called for. The store-and-display function is achieved through the use of the store, display, and set IC keys, the address switches, the data switches, the store/display switch, the scan key, and the CRT display switch. Once the desired intervention is completed, the CPU can be started again.

The normal stopping and starting of the CPU in itself does not cause any alteration in program execution other than in the time element involved in the transition from operating to stopped state.

Machine checks occurring during store-and-display operations do not log immediately but create a pending log condition that can be removed by a system reset or check reset. The error condition, when not disabled, forces a log-out and a subsequent machine check interruption when the CPU is returned to the operating state.

Initial Program Loading

Initial program loading (IPL) is provided for initiation of processing when the contents of storage or the PSW are not suitable for further processing.

Initial program loading is initiated manually by selecting an input device with the load-unit switches and pressing the load key.

Pressing the load key causes a system reset, turns on the load light, turns off the manual light, and initiates a read operation from the selected input device. When reading is completed satisfactorily, a new PSW is obtained, the CPU starts operating, and the load light is turned off.

The system reset suspends all instruction processing, interruptions, and timer updating and also resets all channels, on-line nonshared control units, and I/O devices. The contents of general and floating-point registers remain unchanged.

When IPL is initiated, the selected input device starts transferring data. The first 24 bytes read are placed in storage locations 0-23. Protection, program-controlled interruption, and a possible incorrect length indication are ignored. The double word read into location 8 is used as the channel command word (CCW) for reading more than 24 bytes. When chaining is specified in this CCW, the operation proceeds with the CCW in location 16. Either command chaining or data chaining may be specified.

When the device provides channel end for the last operation of the chain, the I/O address is stored in bits 21-31 of the first word in storage. Bits 16-20 are made zero. Bits 0-15 remain unchanged.

The CPU subsequently fetches the double word in location 0 as a new PSW and proceeds under control of the new PSW. The load light is turned off. No I/O interruption condition is generated. When the I/O operations and PSW loading are not completed satisfactorily, the CPU idles, and the load light remains on.

Programming Notes: Initial program loading resembles a START I/O that specifies the I/O device selected by the load-unit switches and a zero protection key. The CCW for this START I/O is simulated by CPU circuitry and contains a read command, zero data address, a byte count of 24, chain-command flag on, suppress-length-indication flag on, program-controlled-interruption flag off, chain-data flag off, and skip flag off. The CCW has a virtual address of zero.

Initial program loading reads new information into the first six words of storage. Because the remainder of the IPL program may be placed in any desired section of storage, it is possible to preserve such areas of storage as the timer and PSW locations, which may be helpful in program debugging.

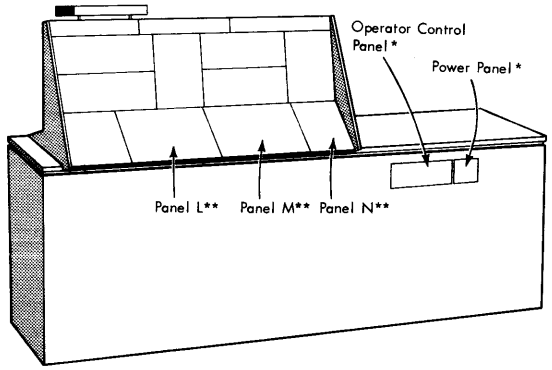
If the selected input device is a disk, the IPL information is read from track zero.

The selected input device may be a channel-to-channel adapter connecting the channels of two CPU's. After a system reset is performed and a read command is issued to this adapter by the requesting CPU, the adapter sends an attention signal to the addressed CPU. That CPU then should issue the write command necessary to load a program into main storage of the requesting CPU.

When the PSW in location 0 has bit 14 set to 1, the CPU is in the wait state after the IPL procedure (the manual, system, and load lights are off, and the wait light is on). Interruptions that become pending during IPL are taken before instruction execution.

Controls

System controls are divided into three logical groups, identified as operator control, operator intervention, and customer engineering control. The grouping of the panels used in performing the operator control and the operator intervention functions is shown in Figure 10. The controls and indicator lights used in operator control are shown in Figure 11, and the controls and indicator lights used in operator intervention are shown in Figure 12.



* Panels containing operator controls. See Figure 11.
** Panels containing operator intervention controls. See Figure 12.

Figure 10. Grouping of Panels on Maintenance Console

Operator Control

The operator-control section of the system control panel contains the controls and indicator lights required by the operator when the CPU is operating under full supervisor control.

Under supervisor control, a minimum of direct manual intervention is required because the supervisor performs operations similar to store and display.

The operator control panel is situated just to the right of and below the main maintenance console panel (see Figure 10). To control another System/360 processor, another set of controls and indicator lights (an optional feature) can be provided on the operator control panel (see Figure 11). One set may be

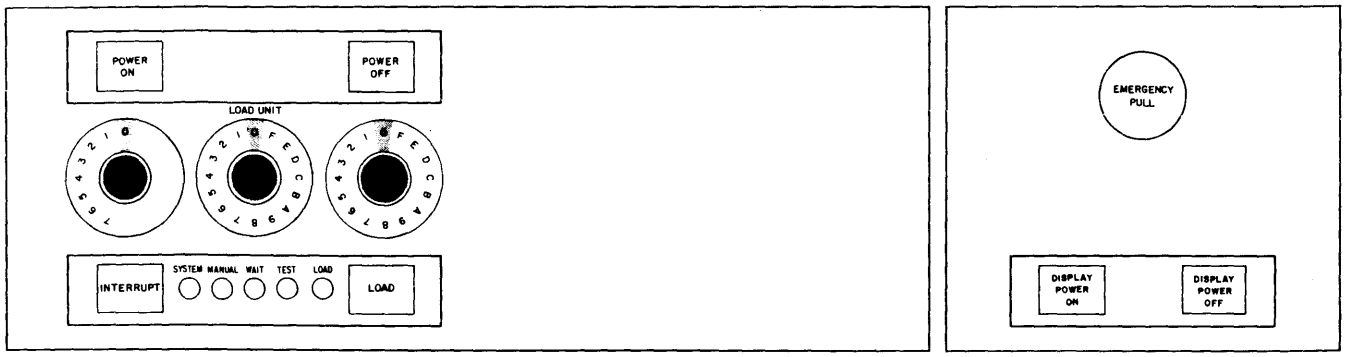


Figure 11. Operator Control Panel and Power Panel

duplicated as a remote panel on a stand-alone operator's console (IBM 2150 Console or IBM 2250 Display Unit Model 1.) This attachment is a standard feature.

The main functions provided by the operator controls are the control and indication of power, the indication of system status, operator-to-machine communication, and initial program loading.

The following table lists (alphabetically) all operator controls and indicator lights and their implementation. (See Figure 11.)

NAME	IMPLEMENTATION
Display Power Off	Key
Display Power On	Key (backlighted)
Emergency Pull	Pull switch
Interrupt	Key
Load	Key
Load	Indicator light
Load Unit	Rotary switches (3)
Manual	Indicator light
Power Off (System)	Key
Power On (System)	Key (backlighted)
System	Indicator light
Test	Indicator light
Wait	Indicator light

Display Power Off

The display-power-off key is pressed to initiate the power-off sequence of the display console integrated with the system control panel. The key is situated on the power panel (Figure 11). The contents of the upper 4,096 bytes of display console buffer storage (containing format control data) are preserved following power off to the display console.

Display Power On

The display-power-on key, situated on the power panel (Figure 11), is pressed to initiate the power-on sequence of the display console integrated with the system control panel. While power is on the display console, the key is backlighted white. The contents of the upper 4,096 bytes of display console buffer storage (containing format control data) are preserved following power on for the display console.

Emergency Pull

Pulling the emergency-pull switch turns off all power beyond the power-entry terminal on every unit that is part of the system or that can be switched onto the system. Therefore, the switch controls the system proper and all off-line and shared control units and I/O devices.

The switch latches in the out position and can be restored to its in position by maintenance personnel only.

Interrupt

The interrupt key is pressed to request an external interruption. The interruption is taken when it is allowed and when the CPU is not stopped. Otherwise, the interruption remains pending. Bit 25 in the interruption-code portion of the current PSW is made one to indicate that the interrupt key is the source of the external interruption. The key is effective when power is on the system.

Load (Key)

The load key is pressed to begin initial program loading. (See "Initial Program Loading" elsewhere in this section.) It is effective while power is on the system.

Load (Light)

The load light is on during initial program loading; it is turned on when the load key is pressed and is turned off after the new PSW is successfully loaded.

Load Unit

The three load-unit switches provide the 11 rightmost I/O address bits of the device to be used for initial program loading.

The leftmost switch has eight positions, labeled 0-7. The other two switches are 16-position switches labeled hexadecimally 0-F.

Manual

The manual light is on when the CPU is in the stopped state. Several of the manual controls are effective only when the CPU is stopped, that is, when the manual light is on.

Power Off (System)

The power-off key is pressed to initiate the power-off sequence of the system. The contents of main storage (but not the keys in storage associated with the protection features) are preserved, provided that the CPU is in the stopped state. The key is effective while power is on the system.

Power On (System)

The power-on key is pressed to initiate the power-on sequence of the system. As part of the power-on sequence, a system reset is performed in such a way that the system performs no instructions or I/O operations until explicitly directed. The contents of main storage are preserved.

The power-on key is backlighted white when power is on the entire system. The key is backlighted red during the power-on sequence and when any remote/local power control switch in the power system is in the local position. If there is a loss of power in some section of the processor, main storage units, or channels, the light will change from white to red. The power-on key is effective only when the emergency-pull switch is in its in position.

System

The system light is on when the CPU-cluster usage meter or customer engineering meter is running. These meters are situated on the left side of the display console.

The states indicated by the wait and manual lights are independent of each other; however, the state of the system light is not independent of the states of the wait and manual lights. The possible conditions when power is on are:

SYSTEM LIGHT	MANUAL LIGHT	WAIT LIGHT	CPU STATE	I/O STATE
Off	Off	Off	*	*
Off	Off	On	Wait	Not working
Off	On	Off	Stopped	Not working
Off	On	On	Stopped, Wait	Not working
On	Off	Off	Running	Undetermined
On	Off	On	Wait	Working
On	On	Off	Stopped	Working
On	On	On	Stopped, Wait	Working

*Abnormal Condition

Test

The test light is on when a manual control is not in its normal position or when a maintenance function is being performed for the CPU, channels, or main storage.

Any abnormal setting of a switch that is on the system control panel or on any separate maintenance panel for the CPU, main storage, or channels and that can affect the normal operation of a program causes the test light to go on.

The test light may be on when certain diagnostic functions are activated or when certain abnormal circuit-breaker or thermal conditions occur.

The test light is not an indication of the state of the marginal voltage controls.

The test light is on when any of the following manual controls is not in its normal position:

Address compare
Address increment (up-by-BSM/beat)
Address increment (1-up store/display)
Block scan
CPU check stop
CRT display and tape operation
Disable interval timer
Enter instruction
Frequency margin range
Inhibit multi-access
Inhibit overlap
Maintenance console test, rotary
MCW active
MCW to PSCE
Rate
Repeat
Reverse CBR PTYS
Storage address alteration
Storage test (EMS-only/MWS-only)
Storage test (store/fetch)

Wait

The wait light is on when the CPU is in the wait state. The wait state exists whenever bit position 14 of the current PSW contains a one. The wait state can be changed to the running state only by loading a new PSW in which bit position 14 contains a zero; it cannot be changed by pressing the system reset key.

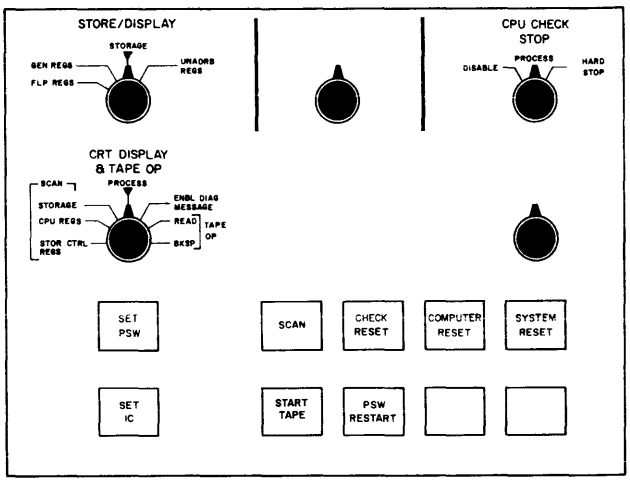
Operator Intervention

This section of the system control panel contains the controls required for the operator to intervene in normal programming operation. These controls are intermixed with the customer engineering controls.

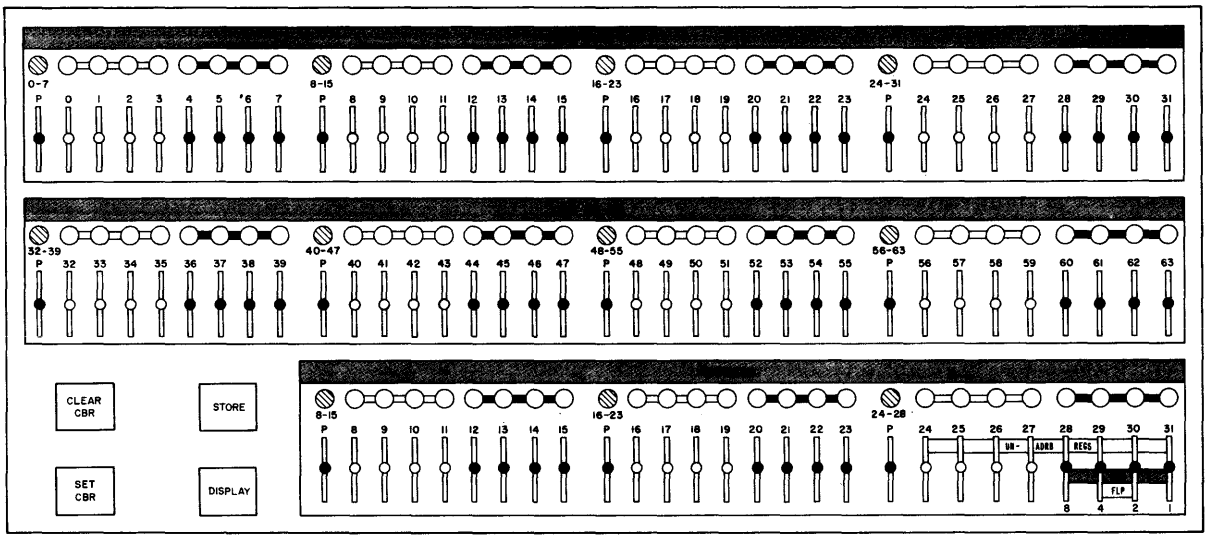
Operator intervention provides the system-reset function and the store-and-display function.

The following table lists (alphabetically) all intervention controls and their implementation. (See Figure 12.)

PANEL N



PANEL M



PANEL L

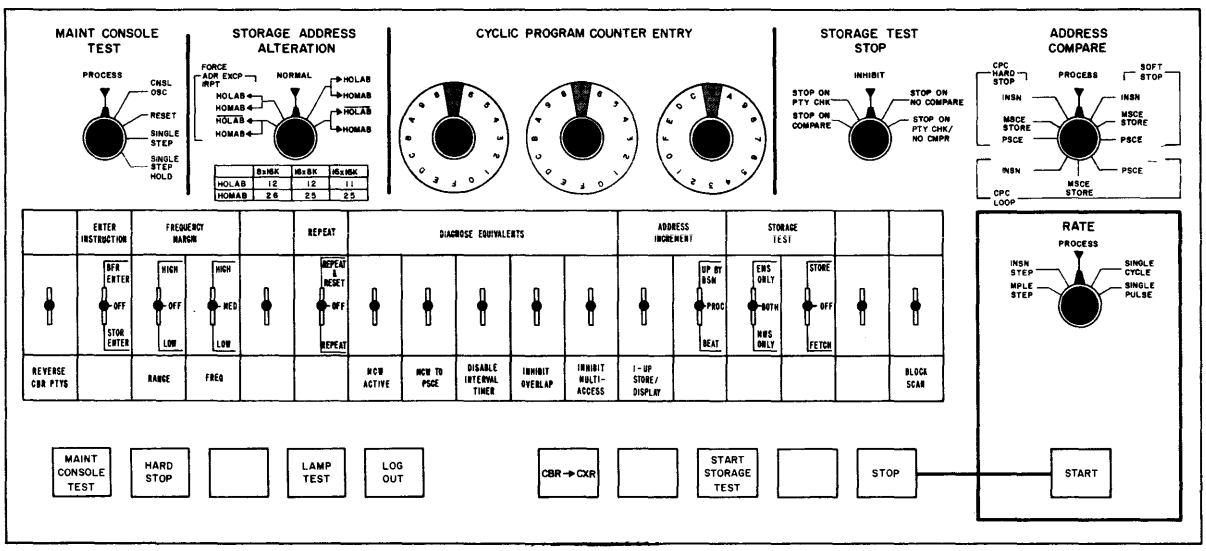


Figure 12. Operator Intervention Controls and Indicator Lights

NAME	IMPLEMENTATION
Address	Key switches
Address Compare	Rotary switch
Address Increment (1-Up Store/Display)	Key switch
Block Scan	Key switch
CRT Display and Tape Operation	Rotary switch
Data	Key switches
Display	Key
Log-Out	Key
PSW Restart	Key
Rate	Rotary switch
Scan	Key
Set IC	Key
Set PSW	Key
Start	Key
Stop	Key
Store	Key
Store/Display	Rotary switch
System Reset	Key

Address

The 27 address switches (24 address switches and three parity switches) are nonlocking key switches and are used to address a storage location or register, as specified by the store/display switch.

The switches have nonlocking set and reset positions; they are in a neutral position when they are not being operated. The associated bit position of the address register is set or reset according to the position to which the switch is operated. (When the switch is operated down, the bit position is *set.*)

The contents of the address register are the output of the address switches and can be altered by the manipulation of those switches or by the address stepping circuitry. The contents of the address register are indicated continuously so that any manipulation of the address switches can be seen.

The three low-order bit positions are not used in main-storage addressing and are not affected by the address stepping circuitry. Main-storage addressing always specifies a double-word boundary. However, in the performance of the address-compare and for register selection, the three low-order bit positions are used.

Parity for each byte is indicated by the parity indicators in the address register and is generated automatically whenever the address register is used. The three parity switches of the 27 switches do not affect usage; when activated, they turn the associated address-register bit on or off, but parity is automatically updated in the address register before the address is used.

Address Compare

The address-compare rotary switch is used to control synchronizing pulses, program loops, and machine stops by means of address comparisons during instruction-fetch or data-store operations. The switch has 10 active positions, seven of which are for customer engineering use. If the switch is in other than the process position, the test light will be on.

The address-compare switch can be manipulated among the three customer settings, described below, without disrupting CPU operation other than by causing the address-comparison stop.

Process: When the switch is in this position, a synchronizing pulse is provided when the address specified in the address register matches the instruction address. The pulse occurs when decoding of the instruction begins and may be used to synchronize an oscilloscope at the start of an instruction.

This position is the normal operating position for the switch.

Insn Soft Stop: When the switch is in this position, the CPU enters the stopped state when the address specified in the address register matches the instruction address. This position may be used to control the stopping point of a program.

The instruction-fetch operation, all other outstanding operations, and all pending interruptions that are allowed are completed before the CPU enters the stopped state.

MSCE Store Soft Stop: When the switch is in this position, the CPU enters the stopped state when the address specified in the address register matches a main-storage or extended-main-storage address specified in any CPE store operation or in an I/O store operation into high-performance main storage.

The store operation, all other outstanding operations, and all pending interruptions that are allowed are completed before the CPU enters the stopped state.

The PSCE soft stop setting, the CPC hard stop settings, and the CPC loop settings are for customer engineering use.

Address Increment (1-Up Store/Display)

When the store/display switch is in the storage position, the setting of the 1-up store/display switch determines which double-word location in storage is accessed.

Off: When the switch is in this position (centered), pressing the store key or the display key causes the double-word storage location to be accessed that is referenced by the maintenance console address register.

On: When the switch is in this position (down), pressing the store key or the display key causes the double-word storage location to be accessed that is referenced by the maintenance console address register. After the access is made, the maintenance console address register is automatically incremented by one double word.

NOTE: When the 1-up store/display switch is on, the address increment (up-by-BSM/beat) key switch must be in the process position.

Block Scan

The block scan key switch is operated to inhibit the scanning of registers or main storage during a CRT-display operation.

Off: When the switch is in this position (centered), pressing the scan key initiates the normal scan function (see the description of the scan key).

On: When the block scan switch has been placed in this position (down), pressing the scan key immediately after setting the block scan switch causes the information displayed during the last storage display operation to be displayed again; the data specified by the CRT display switch and related controls are not scanned, and no data are transferred to the buffer storage of the display console.

When the switch is in the on position, the test light will be on.

CRT Display and Tape Operation

The CRT-display portion of this switch, identified as "Scan," is used to connect the display console with either an I/O channel or the system control panel. When it connects the display console to the system control panel, the switch is used to determine the type of display to be produced on the CRT when the scan key is pressed.

The tape operation portion of this switch is for customer engineering use.

When the switch is in any position other than process, the test light will be on.

Process: When the switch is in this position, the display console is connected to either a 2860 channel or a 2870 selector subchannel and is under program control.

Storage: When the switch is in this position, pressing the scan key initiates an operation in which, beginning at the address indicated by the address register, 16 double words are fetched from storage (one double word at a time), placed in the console buffer register, and then transferred to the buffer storage of the display console; these 16 double words are then displayed on the CRT in hexadecimal form with the storage address of each double word.

NOTE: When the switch is in the storage position, the address increment (up-by-BSM/beat) switch must be in the process position.

Pressing the scan key again causes the next 16 double words in storage to be fetched and displayed.

When the block scan switch is operated, the fetch-and-transfer operation is inhibited. Pressing the scan key immediately after the block scan switch has been operated causes the 16 double words displayed during the last storage display operation to be displayed again.

CPE Regs: When the switch is in this position and the CPU is in the stopped state, pressing the scan key initiates an operation in which the contents of the CPE registers are placed, in order, into the console buffer register and then transferred to pre-assigned buffer storage areas of the display console; then this data and the identifications of the registers are displayed on the CRT.

Operating the block scan switch immediately before pressing the scan key inhibits the fetch-and-transfer operation; when the scan key is then pressed, data displayed during the last CPE-register-display operation are displayed again.

Stor Ctrl Regs: When the switch is in this position and the CPU is in the stopped state, pressing the scan key initiates an operation identical to that described for the CPE-regs position, except that the registers of the storage control areas are scanned and displayed.

Operating the block scan switch immediately before pressing the scan key inhibits the fetch-and-transfer operation; when the scan key is then pressed, data displayed during the last storage-control-register-display operation are displayed again.

Data

The 72 data switches (64 data switches and eight parity switches), labeled CBR, are nonlocking key switches and are used to enter data into selected areas of the CPU or storage.

The contents of the console buffer register (CBR) are normally the output of the data switches; the contents of this register are altered either by manipulation of these switches or by a storage fetch operation.

The switches have nonlocking set and reset positions; they are in a neutral position when they are not being operated. The associated bit position of the buffer register is set or reset according to the position to which the switch is operated. (When the switch is operated down, the bit position is *set.*) The contents of the console buffer register are indicated continuously so that any manipulation of the data switches can be seen.

Data are stored according to the setting of the store/display switch and the contents of the address register. The actual operation of storing is initiated by pressing the store key. The parity is automatically generated whenever the data are transferred.

Display

The display key is pressed to place data into the console buffer register, as determined by the setting of the store/display switch and by the contents of the address register. The lights for the console buffer register continuously display the contents of that register.

When the designated location is not available, the displayed information is unpredictable.

This key is effective only when the CPU is in the stopped state.

Log-Out

Pressing the log-out key causes a log-out of machine status into main storage. The log-out area is 1,520 bytes (190 double words); it begins at location 128_{10} and continues through location 1647_{10}.

The key is effective only when the CPU is in the stopped state.

NOTE: Machine status is also logged out when a machine-check interruption takes place.

PSW Restart

The PSW-restart key is pressed to initiate the following operations in sequence:

1. System reset.
2. Loading of a new PSW from location 0.
3. Instruction fetching starting at the new program location specified by the new PSW.
4. Execution of instructions as specified by the setting of the rate switch.

Rate

The rate rotary switch is used to indicate the way in which instructions are to be performed. The test light is on if the rate switch is set to any position other than process.

The position of the rate switch should be changed only while the CPU is in the stopped state. Otherwise, results are unpredictable.

Process: When the switch is in this position, the system operates at normal speed after the start key is pressed. The decoding of instructions is halted by pressing the stop key.

Insn Step: When the switch is in this position, one instruction is completely executed each time the start key is pressed. The CPU automatically halts in the stopped state. When the start key is pressed but before the one instruction is processed, interruptions that were allowed but became pending during the stopped state are processed prior to the execution of the next instruction.

Mple Step: When the switch is in this position, an instruction is executed every 100 milliseconds for as long as the start key is pressed. The CPU automatically halts in the stopped state when the start key is released.

Single Cycle: This position is for customer engineering use.

Single Pulse: This position is for customer engineering use.

Scan

The scan key is pressed to produce a display on the cathode-ray tube of the display console. The display that is produced is determined by the setting of the CRT display switch.

Whether the display represents updated information depends on whether the block scan switch has been operated. Pressing the scan key immediately after the block scan switch has been placed in the operating position causes the information displayed during the last storage display operation to be displayed again.

This key is effective only when the CPU is in the stopped state.

Set IC

The set IC (instruction counter) key is pressed to enter the contents of bit positions 40-63 of the console buffer register into bit positions 40-63 (the instruction address part) of the current PSW.

This key is effective only when the CPU is in the stopped state.

Set PSW

The set PSW (program status word) is pressed to enter the contents of bit positions 0-15 and 32-63 of the console buffer register into bit positions 0-15 and 32-63 of the current PSW.

The key is effective only when the CPU is in the stopped state.

Start

The start key is pressed to start instruction execution as defined by the setting of the rate switch.

Pressing the start key after a normal halt causes instruction processing to continue as if no halt had occurred, provided that the rate switch is in the process, instruction-step, or multiple-step position.

Pressing the start key after system reset without first having introduced a new instruction address yields unpredictable results.

Pending interruptions that are allowed will be honored before the first instruction is executed.

The key is effective only while the CPU is in the stopped state.

Stop

The stop key is pressed to terminate machine operation without destroying the machine environment. The CPU enters the stopped state after all instructions previously decoded have been executed, after all pending interruptions have been processed, and after an interruption that became pending while the CPU was in the decode or stop-decode state has been processed.

As the CPU enters the stopped state, the manual light goes on. After stopped state has been entered, no interruptions are processed.

The stop key is active while power is on the system.

Store

The store key is pressed to store data from the console buffer register into the location specified by the setting of the store/display switch and by the contents of the address register.

Store protection is ignored. When the location designated by the address register and by the setting of the store/display switch is not available, no data are stored.

The key is active only when the CPU is in the stopped state.

Store/Display

The store/display rotary switch is used to specify the part of the system that is to be stored into or displayed, according to the contents of the address register and by the use of the store or display key.

Storage: When the switch is in this position and the CPU is in the stopped state, the data at the address indicated by the address register are placed in the console buffer register when the display key is pressed or are replaced by the contents of the console buffer register when the store key is pressed.

Unadrbl Regs: When the switch is in this position and the CPU is in the stopped state, data in a register (in the CPE or in the storage control units) whose pseudo-address is in the address register can be placed in the console buffer register by pressing the display key.

Gen Regs: When the switch is in this position and the CPU is in the stopped state, the contents of the general register (indicated by the address register) can be placed in the console buffer register by pressing the display key or can be replaced by the contents of the console buffer register by pressing the store key. The contents of the general register are displayed left-justified in the console buffer register; for store operations, the data must be placed in the left half of the buffer register.

FLP Regs: When the switch is in this position and the CPU is in the stopped state, the contents of the floating-point register that is indicated in the address register are placed in the console buffer register by pressing the display key or are replaced by the contents of the console buffer register by pressing the store key.

System Reset

The system-reset key is pressed to reset on-line channels, control units, and CPU controls to their initial states. All check indicators are reset. The current PSW, data flow registers, keys in storage, and main storage are not reset. The CPU is placed in the stopped state, and all pending interruptions are eliminated. The reset function does not affect any off-line or shared devices.

This key is active while power is on the system.

Key Switch and Meters

The customer usage meter for the CPU cluster and the customer engineering (CE) meter for the CPU cluster are both situated on the left side of the display console.

The Model 91 CPU cluster includes the processor, the maintenance console, the CPU power supplies, the power distribution unit, and the main storage.

A key switch controls which meter is to run. If the key switch is in the customer-operation position, the usage meter accumulates time when the system does productive work. If it is in the CE position, the CE meter accumulates time instead, on the same basis as the usage meter it replaces.

Appendix A: Coding Considerations

Although the Model 91 performs CPU operations in a highly parallel fashion, no elaborate optimization plan need be followed in preparing programs for CPU processing. They may, for the most part, be written in a straightforward IBM System/360 code. However, if it has been determined that a program will benefit by some modification, the following suggestions may prove helpful.

1. Attempt to minimize the use of the following instructions because they are executed by means of interpretive program subroutines: ADD DECIMAL, COMPARE DECIMAL, DIVIDE DECIMAL, MULTIPLY DECIMAL, SUBTRACT DECIMAL, and ZERO AND ADD.

2. Place index loading and incrementing instructions well ahead of instructions that use them for address generation. In a loop, a convenient place for an indexing instruction such as ADD (AR) is at the end of the loop, just before a BRANCH ON INDEX LOW OR EQUAL; by the time the branch is completed, the index registers will be ready for use.

3. The instructions LOAD ADDRESS, BRANCH ON COUNT (BCT, BCTR), BRANCH ON INDEX LOW OR EQUAL, and BRANCH ON INDEX HIGH all use the address adder to change a general register. As suggested in rule 2, make sure that the registers required are available.

4. The LOAD ADDRESS instruction requires three cycles that cannot be overlapped; it is also subject to delays if registers are unavailable. Instructions such as ADD (A, AR) require only one unoverlapped cycle and are not subject to delays if registers are unavailable. In most cases, therefore, the LOAD ADDRESS instruction should be replaced by an AR instruction. There are some situations when the LOAD ADDRESS instruction is preferable:

   a. When the register to be used is needed for addressing by the very next instruction.
   b. When the fixed-point execution element is busy with a lengthy instruction sequence and a register is needed for addressing within the next few cycles.
   c. When the condition code must not be changed.

5. Because the Model 91 fetches and stores double words, align operands for VFL instructions on double-word boundaries for faster operations.

6. In normal coding, a condition-setting instruction immediately precedes most BRANCH ON CONDITION instructions. On the Model 91 it is usually advantageous to place neutral instructions, such as those dealing with loads and stores, between the condition-setting instruction and the conditional branch.

7. Avoid storing into the next several words of the instruction stream.

8. Whenever possible, contain a loop in the instruction stack in such a way that it will be executed in loop mode (see the discussion of loop mode in "Instruction Processor").

9. Because all instructions that store data use the same three store address registers and the same three store data buffers, if a fourth store is encountered before a store address register is freed, the instruction processor must wait. In some cases it is possible to keep from having more than three stores in a row. For example, if it is necessary to store data from six registers by using one STORE MULTIPLE instruction, only three SAR's are required if the first address started on a double-word boundary; four stores are required otherwise.

10. When only two registers are to be loaded, using two LOAD (L) instructions is faster than using a LOAD MULTIPLE instruction. However, when four or more registers are to be loaded, the LOAD MULTIPLE instruction is preferable. Also, the STORE MULTIPLE instruction is usually better than repeated STORE (ST) instructions because it requires fewer SAR's and SDB's.

11. Repeated accesses to different double words in the same storage module cause conflicts. Repeated accesses to the same double word, however, usually cause no conflicts because of the combined-accessing feature in the main storage control element (see the discussion of multiple CPE requests to storage in "Main Storage Control Element").

Appendix B: Timing Considerations

For other models of the IBM System/360, average times can be given for each instruction. For a parallel system such as the Model 91, however, no average times are meaningful because the amount of overlap varies from program to program.

The following information is relevant to any considerations of timing. It is presented to give the reader an appreciation of some of the major aspects of timing in the Model 91 but is not intended to be comprehensive. In the discussion, the term "cycle" is used to refer to a major-machine-cycle time of 60 nanoseconds.

Conditions That Delay the Instruction Processor

Any of the following conditions will delay the instruction processor:

1. The next instruction is unavailable.
2. The machine is in conditional mode, and the next instruction is an instruction to be executed by the instruction processor or is a variable-field-length instruction. (An unconditional branch or a no-operation instruction, however, can be executed in conditional mode.)
3. A general register is unavailable for addressing for the next instruction.
4. A general register is unavailable for modification by the next instruction — a condition that applies only to an instruction-processor instruction, such as LOAD ADDRESS or BRANCH ON INDEX LOW OR EQUAL, which changes a general register.
5. The next instruction requires an address generation, but a previous instruction will not be able to complete its address generation for another cycle.
6. The next instruction requires a fixed-point buffer register, but all fixed-point buffer registers are busy.
7. The next instruction requires a floating-point buffer register, but all floating-point buffer registers are busy.
8. The next instruction is a fixed-point operation, but the fixed-point operation stack is full.
9. The next instruction is a floating-point operation, but the floating-point operation stack is full.
10. The next instruction requires a store, but all store address registers are busy.
11. An instruction is decoded whose execution is delayed until the completion of all previously decoded instructions.

Important Transmission Time

Any of the following transmissions requires one cycle. In most instances these transmissions take place concurrently with other operations, but there may be instances in which delays due to these transmissions will directly affect the timing.

1. A fixed-point or floating-point operation from the instruction processor to the fixed-point operation stack or the floating-point operation stack, respectively.
2. An activate or cancel signal from the instruction processo· to the fixed-point operation stack or the floating-point operation stack.
3. A condition-code indication from an execution unit to the instruction processor.
4. A general-register-available indication from the fixed-point execution element to the instruction processor.
5. A buffer-free indication from the fixed-point execution element or the floating-point execution element to the instruction processor.
6. An operation-stack-position-free indication from the fixed-point execution element or the floating-point execution element to the instruction processor.
7. A store-address-register-free indication from the main storage control element to the instruction processor.

Branches

When loop mode is not set, the first cycle of a branch is the usual decoding in the instruction processor. The next two cycles are address generations for the target and target + 1 double words; the two temporary fetches are initiated immediately after the address generations. The minimum time for any branch out of the instruction stack, therefore, is two cycles plus the access time.

The test for a conditional branch is normally made after the address generation. There are two types of conditional branches: those whose condition is set by the instruction processor and those whose condition is set by the fixed-point or floating-point execution element. For the instructions BRANCH ON COUNT (BCT, BCTR), BRANCH ON INDEX HIGH, and BRANCH ON INDEX LOW OR EQUAL, the condition is set by the instruction processor. For the BRANCH ON CONDITION (BC, BCR) instruction, the condition is set by the execution ele-

ments. (Masks of 0 and 15 are special cases and are detected during the decoding cycle.)

When the condition is set by the instruction processor, no further instructions are decoded until all tests have been completed. The following are instruction processor times (in cycles) for some of the more important branches:

	TARGET IN STACK		TARGET NOT IN STACK	
	LOOP MODE	QUICK LOOP	LOOP MODE	NOT LOOP MODE
BX, Branch	4	3	6 + access time	8 (or 2 + access time)*
BX, No Branch	6	5	5	6
BCT, Branch	4	3	5 + access time	7 (or 2 + access time)*
BCT, No Branch	5	4	4	5

*The actual time required is the longer of the two times listed.

When the condition is set by an execution element, the first three cycles of the branch are taken by the instruction processor, and the temporary fetches are made. The instruction processor then enters conditional mode until the condition code is determined.

In conditional mode, no additional instruction fetches are made. The instruction processor continues to decode instructions, generate addresses, and issue operations to the fixed-point operation stack and the floating-point operation stack; the operations are conditional and cannot be decoded or executed until an activate signal is sent by the instruction processor.

The instruction processor continues to decode instructions conditionally until any of the following conditions occurs:

1. The condition code is set.
2. No more instructions are available in the stack.
3. The fixed-point or floating-point operation stack is filled.
4. An instruction-processor or variable-field-length instruction is encountered (except for an unconditional branch or a no-operation instruction, which can be executed in conditional mode).

When the condition code is set, the instruction processor takes one cycle to make a decision. If the branch is not taken, an activate signal is sent to the fixed-point and floating-point operation stacks, and the instruction processor continues decoding instructions. If the branch is taken, a cancel signal is sent to the fixed-point and floating-point operation stacks and to the SAR's, and the instruction processor begins decoding instructions along the new path. When conditional mode is ended, instruction fetching resumes along the correct path.

When the machine is in loop mode, no temporary fetches are made for conditional branches.

An unconditional branch (BC 15 or BCR 15) takes either six cycles or two cycles plus the access time. A branch within the stack takes five cycles, and a branch closing a loop takes two cycles.

The BRANCH AND LINK instructions (BAL, BALR) require four cycles plus the time required for access or plus the time required for the condition code to be determined, whichever is longer. The BRANCH AND LINK instruction destroys loop mode.

A no operation (BC 0,X; BCR 0,R; BCR C,0) requires one cycle, a count without branching (BCTR R,0), three cycles; a link without branching (BALR R,0), five cycles or the time until the condition code is determined; and an EXECUTE, five cycles plus the access time plus the target execution time.

Fixed-Point Execution

The following information is pertinent to fixed-point execution timing:

1. Decoding proceeds serially.
2. No conditional operation can be decoded until it has been activated or canceled.
3. Canceled operations are decoded in one cycle.
4. An active operation is not completely decoded until the cycle before its execution starts.
5. Execution can begin if the following conditions have been met:
   a. The operation has been decoded.
   b. The data are available.
   c. The execution circuitry is free.
6. As soon as the decoding is completed for a 1-cycle operation, the instruction processor is notified that the stack position is free. For operations of more than 1 cycle, the stack-position-free notification is delayed until the second or third cycle. Notification that the fixed-point buffers are released is given to the instruction processor during the first cycle for all instructions except CONVERT TO BINARY and DIVIDE (D), which do not release the buffers until during the last cycle.

Floating-Point Execution

The following information is pertinent to floating-point execution timing:

1. Decoding proceeds serially.
2. No conditional operation can be decoded until it has been activated or canceled.
3. Canceled operations are decoded in one cycle.
4. Operations that do not require an execution unit can be decoded in one cycle.
5. Operations that require an adder or a multiplier can be decoded in one cycle if a reservation station is available.
6. If a decode is waiting for a reservation station, it can be completed on the cycle before the result of that reservation station goes on the common data bus.

7. Precision conflicts (differences in precision between two overlapped floating-point operations using the same floating-point register) and RR instructions for which both registers are free may cause exceptions to the preceding rules.

8. The test for LOAD AND TEST (LTDR and LTER) is made during the common data bus cycle.

9. An operation in which the adder is used can begin if the following conditions have been met:
   a. The operation has been decoded.
   b. The data are available.
   c. Another add with higher priority is not beginning on the same cycle.
   d. The execution circuitry is free.

10. A multiply or divide can begin if the following conditions have been met:
   a. The operation has been decoded.
   b. The data are available.
   c. Another multiply or divide with higher priority is not beginning on the same cycle.
   d. The execution circuitry is free.

11. If more than one unit requests the common data bus simultaneously, the following operations are given priority in the order indicated: loads, adds, multiplies.

12. As soon as an operation has been decoded, the instruction processor is notified that the stack position is free.

13. If the operation has already been decoded, the buffer is set free as soon as the data enter it.

14. If the buffer is filled before the operation is decoded, the buffer is set free one cycle after the decoding.

Selected Execution Times

Because of the concurrency achieved in the Model 91, the effective time required by a given instruction is not directly related to the rate at which that instruction can be processed.

The following is a list by category of the number of cycles required by the appropriate execution element to process certain instructions. The times listed do not include any of the other processing time required for that instruction and do not reflect the effects of simultaneous operations or overlap. Instructions are listed by their mnemonics.

	NO. OF MACHINE CYCLES
Fixed-Point Instructions	
A, AH, AL, ALR, AR, C, CH, CL, CLR, CR, IC, L, LCR, LH, LNR, LPR, LR, LTR, N, NR, O, OR, S, SH, SL, SLR, SR, ST, STC, STH, X, XR	1
SLA, SLL, SRA, SRL	2
SLDA, SLDL, SRDA, SRDL	3-4
MH	7
M, MR	7-11
D, DR	36-37
CVB	17-18
CVD	17-32
Immediate Instructions	
Fetch only: CLI, TM	1
Store only: MVI	1
Fetch and Store: NI, OI, XI	2
*Floating-Point Instructions**	
LD, LDR, LE, LER, LTDR, LTER, STD, STE	0
AD, ADR, AE, AER, AU, AUR, AW, AWR, CD, CDR, CE, CER, HDR, HER, LCDR, LCER, LNDR, LNER, LPDR, LPER, SD, SDR, SE, SER, SU, SUR, SW, SWR	2
MD, MDR, ME, MER (normalized numbers)	3
MD, MDR, ME, MER (unnormalized numbers)	4
DE, DER	9
DD, DDR	12

* The 0-cycle instructions do not require an execution unit. The 2-cycle instructions are executed in the adder. The 3-, 4-, 9-, and 12-cycle instructions are executed in the multiplier.

Index

Where more than one page-reference is given, major references appear first and in italic type.

READER'S COMMENT FORM

IBM System/360 Model 91
Functional Characteristics

GA22-6907-3

If you desire a reply by the group that prepared this manual, include your name and address.

From

NAME _____ OFFICE/DEPT NO. _____

CITY/STATE _____ ZIP CODE _____ DATE _____

- How did you use this publication?

  ☐ As a reference source ☐ As a classroom text ☐ As a self-study text

We would appreciate your comments; please give section or figure titles where appropriate.

- What sections or figures were particularly useful or understandable to you?

- What sections or figures could be improved?

- What sections or figures require additional information?

- Any other comments?

- How do you rate this manual?

Thank you for your cooperation. WTC users must add postage.

GA22-6907-3

YOUR COMMENTS, PLEASE . . . . . .

This SRL manual is part of a library that serves as a reference source for systems analysts, programmers and operators of IBM systems. Your comments will help us produce better publications for your use. Each reply will be carefully reviewed by the persons responsible for writing and publishing this material. All comments and suggestions become the property of IBM.

Note: Please direct any requests for copies of publications, or for assistance in using your IBM system, to your IBM representative or to the IBM branch office serving your locality.

fold fold

FIRST CLASS
PERMIT NO. 419
POUGHKEEPSIE, N.Y.

BUSINESS REPLY MAIL
NO POSTAGE STAMP NECESSARY IF MAILED IN THE UNITED STATES

POSTAGE WILL BE PAID BY . . . . . .

IBM CORPORATION
P.O. BOX 390
POUGHKEEPSIE, N.Y. 12602

ATTENTION: CUSTOMER MANUALS, DEPT. B98

fold fold

IBM

International Business Machines Corporation
Data Processing Division
1133 Westchester Avenue, White Plains, New York 10604
(U.S.A. only)

IBM World Trade Corporation
821 United Nations Plaza, New York, New York 10017
(International)

CUT ALONG THIS LINE

IBM System/360 Model 91 Functional Characteristics Printed in U.S.A. GA22-6907-3

IBM System/360 Model 91 Functional Characteristics Printed in U.S.A. GA22-6907-3

International Business Machines Corporation
Data Processing Division
1133 Westchester Avenue, White Plains, New York 10604
(U.S.A. only)

IBM World Trade Corporation
821 United Nations Plaza, New York, New York 10017
(International)