Multiprocessing hardware as implemented on System/370 is presented. With emphasis on tightly-coupled systems configurations, topics include instructions and facilities, and a comparison with prior IBM multiprocessors.

Advanced function extended with tightly-coupled multiprocessing

by R. A. MacKinnon

The availability of Advanced Function Extended brings new hardware to System/370 and, through the Operating System/Virtual Storage 2 (OS/VS2) Release 2, enables users to implement tightly- or loosely-coupled multiprocessing. *Tightly-coupled multiprocessing*, is available only on System/370 Models 158 MP or 168 MP and employs the sharing of a real main storage by two CPUs capable, usually, of executing any of the programs and, depending upon channels, accessing any of the Input/Output (I/O) devices in the system. A single copy of OS/VS2 in shared main-storage controls the entire hardware configuration and manages main storage using Dynamic Address Translation (DAT) and virtual storage. This mode of operation is called *multiprocessor mode*. Through manual configuration switches, each of the 158 or 168 CPUs can be given real storage and I/O; hence each can operate in a completely independent mode from its "peer" processor. In this mode of operation, called *uniprocessor mode*, a separate copy of VS2 resides in each main storage and controls a single, partitioned CPU.

Loosely-coupled multiprocessing is available on any System/370 configuration capable of running OS/VS2 Release 2 and above (Models 145, 155-II, 165-II, 158, 168, and 158/168 MP). A component of OS/VS2 Release 2, called Job Entry Subsystem 3 (JES3), assumes the role of coordinator and controls the flow of work through the system. With loosely-coupled multiprocessing, each CPU stands by itself with its own complement of main storage. A CPU designated the *global* processor is connected to each of the other CPUs by the hardware Channel-to-Channel

Adapter (CTCA) and a shared direct access file. The other CPUs are designated *local* processors and may also interconnect with one another through CTCAs to provide backup if the global processor fails. All jobs enter the system through the global processor including Remote Job Entry (RJE) input. The global processor then performs job-entry services for all processors (including itself). Each CPU system (global or local) contains a copy of OS/VS2, the JES3 subsystem, and has a complement of I/O unless the installation elects to share or pool devices between CPUs. Normally, all unit-record I/O is assigned to the global processor because all work (job streams and systems output) enters and leaves through this CPU. Control information is communicated between global and local processors over the channel-to-channel connection; job-related data is exchanged over channels and through the shared JES3 direct access device that all the processors access. System output (such as operator messages, local printing/punching and RJE output) is returned through the shared direct access device(s) to the global processor which drives the consoles, unit record devices, and RJE terminals (when present). Any or all of the CPUs in a JES3 environment may be tightly-coupled multiprocessing systems. An added measure of flexibility is possible since any of the nonglobal processors may be under the control of Asymmetric Multiprocessing System (ASP) Version 3 under OS/MVT or OS/VS2 Release 1. Thus the ASP-controlled CPUs may be System/360s, System/370 in basic control mode, or System/370 in extended control mode; all are referred to as *mains*.

Implementation of tightly- and loosely-coupled multiprocessing (MP) is under OS/VS at Release 2 or above. Major architectural and internal design changes have been written into the OS/VS2 System Control Program (SCP) which also apply to System/370 CPUs operating under it in uniprocessor configurations—that is, no multiprocessing. The following discussion is a brief overview of three major design innovations in VS2, available to uniprocessors, which firmly establish the foundation upon which multiprocessing is built:

- Parallelism
- Locks
- Reliability/protection

From this, the reader should see that the basic VS2 system can accommodate multiprocessing on a standard basis; multiprocessing is not an "add-on" as with OS/MVT. More detail on OS/VS2 Release 2 multiprocessing components can be found elsewhere in this issue.[1]

parallelism

Many more areas of the OS/VS2 SCP operate multiprogrammed than is the case with OS/MVT or OS/VS2 Release 1. Thus with

Release 2 and above, control program execution derives the same benefits that multiprogramming extends to application programs, but with even greater impact on systems responsiveness and performance because the SCP is the central, controlling resource. Bottlenecks have also been removed from OS/VS2 Release 2 by allowing similar functions to operate in parallel rather than in the serial fashion of MVT or OS/VS2 Release 1. For example, device allocation has been redesigned to allow multiple allocation requests to be under way and executing simultaneously wherever possible. In the past, for instance, a tape could not be allocated while a disk was being allocated. Another approach toward accomplishing the objective of parallelism involves the elimination of previous and known areas of contention such as the job queue (where jobs to be initiated are stored) and the systems catalog. Job queue information is dispersed among the application regions and spool data set; the catalog is implemented as a VSAM data set which results in a faster search with less contention. Finally, the control program spends less time operating in disabled mode (with all interruptions masked off). In this context, less disablement holds significant promise for a more responsive systems environment.

locks

Along with increasing the extent to which systems code operates in parallel, OS/VS2 also takes necessary steps to prevent uncontrolled access to critical systems areas and data that, at specific points in time, cannot be shared. A combination of program structure and Advanced Function Extended hardware are the tools used by OS/VS2 to accomplish this. A system of programmed switches or *locks* is used by OS/VS2 to enqueue one processor on a critical resource currently being utilized by the other CPU. This same technique applies to work within a uniprocessor environment. A key hardware element called *interlocking* prevents simultaneous accessing of storage areas from more than one processor. The new swapping instructions (discussed in a later section) set and test the locks. Locking is important in VS2 not only in a heavily paralleled control program executing on a uniprocessor, but is also an absolute necessity in an MP environment where two processors share, access, and execute SCP code on an interruption-driven basis. VS2 provides the necessary protection in the MP environment while placing major emphasis on not "shutting out" inter-CPU communication or dependent functions (such as executing a spin loop and lock). Lock design also ensures that VS2 does not become "deadlocked" if an expected or awaited unlocking event does not occur.

reliability/ protection

With Release 2, OS/VS2 uniprocessor support provides the basis for a higher level of systems integrity upon which multiprocessing (MP) is built. Several architectural changes within VS2 accomplish this. Unlike OS/MVT and OS/VS2 Release 1 in which the

supervisor and other systems functions all run under protect key 0, in OS/VS2 at Release 2 some significant parts of the control program run in a nonzero key. Effectively a hierarchy of SCP storage protection is implemented utilizing hardware storage-protection keys 0–7; OS/VS2 is thus protecting itself from itself to a far greater degree than in past implementations of OS.

Additionally, a hierarchy or structure has been established within OS/VS2 to contain abnormal termination conditions to the smallest possible amount of code and to handle them by the lowest possible functional recovery routine. Not only does this offer greater opportunity to refresh damaged code, but it is also aimed at terminating, if possible, only the damaged task(s) without affecting the undamaged ones and, in the process, without preoccupying the whole of OS/VS2 in this chore.

Lastly, at Release 2, OS/VS2 resolves the problem that previously existed whereby the application programmer could get control in storage-protection key 0 or "crash" the system if OS were given improper addresses or addresses beyond that user's authorized storage space. One technique used by OS/VS2 to ensure integrity is to respond to many service requests by an application program (for instance, WTOR or GET a record) by performing the requested work not in key 0 but using the key (non-zero) of the requestor. In so doing, VS2 isolates other users and the entire system from consequences of incorrect addresses. The new Program Status Word key handling instructions facilitate implementation of the isolation (discussed later in this paper). Also, each OS/VS2 region and user have their own exclusive virtual address space and are thus protected by the DAT facility through their own complement of segment and page tables. No user can address another user's space and, therefore, both store and fetch protection results.

Figure 1 Layout of main storage

The remainder of this paper discusses in detail the Advanced Function Extended features related to tightly-coupled (shared main-storage) multiprocessing. The topics presented are:

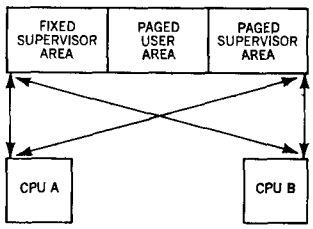
- Shared main-storage MP.
- MP hardware architecture including new instructions and facilities.
- A comparison with prior IBM multiprocessors.

Shared main-storage multiprocessing

Without going into the hardware detail of the next section, this section strives to present an appreciation of the workings of a tightly-coupled multiprocessing system. Here the question becomes: "What is going on in such a system?"

The layout of main storage contains a single copy of the OS/VS2 Release 2 control program (Figure 1), and thus does not appear very different from uniprocessor schematics of main storage. What changes in MP configurations is the number of processors driving the work active in the system and utilizing real storage. Two processors—two Model 158 MP CPUs or two Model 168 MP CPUs—are used to execute code under control of a single copy of OS/VS2 in a virtual storage, multiple virtual address space environment. Whereas uniprocessors have only one task "active" at a time and only one instruction from it being executed, an MP configuration can actually have two active tasks dispatched and each CPU can be independently executing instructions. Each CPU has its own complete set of hardware facilities, channel paths to I/O, and, from its standpoint, complete access to all main storage without any concern to the existence of the other CPU or its activities and thereby providing a single system image to the installation. The layout of main storage appears to either CPU as shown in Figure 2. Because there is a single copy of VS2 and each CPU is "logically" oblivious to the other's presence (although VS2 is aware), a major MP characteristic is the ability of one of the CPUs, given proper signaling and coaching, to assume in most situations the total processing load if the other CPU should drop out for any reason. In this case, the MP configuration not only has the attribute of two processors to apply to task execution in parallel, but also a redundant resource (a whole CPU) to prevent total system stoppage when one CPU ceases processing.

Figure 2 Systems overview

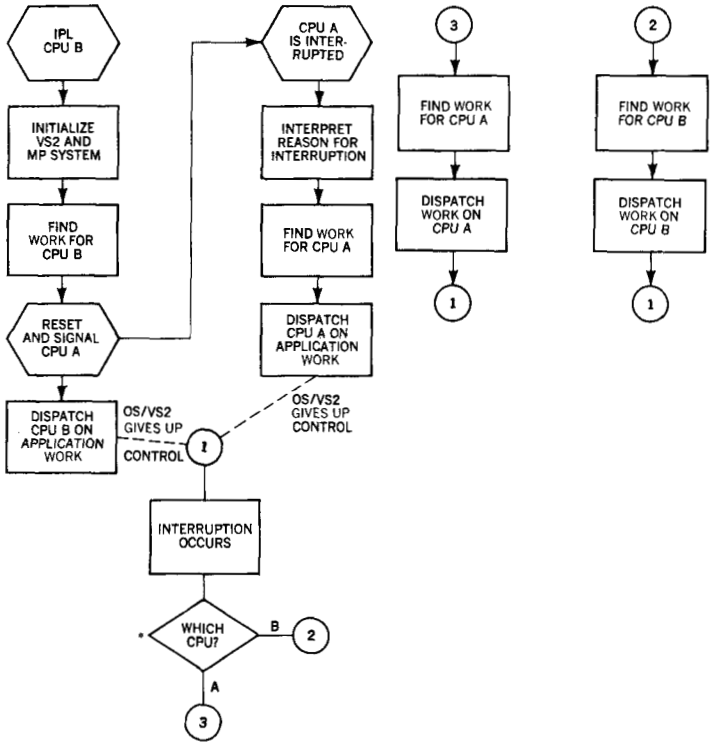


OS/VS2 viewpoint

Another vantage point from which to view shared main-storage multiprocessing is to observe the activities of OS/VS2 and how it functions. A simplified example is illustrated in Figure 3.

The console operator presses the IPL key on one of the CPUs (assume CPU B). This processor proceeds to initialize OS/VS2 in main storage and perform basic housekeeping necessary to bring up the entire MP system. CPU B, having finished this start-up, now finds both itself and CPU A ready to undertake processing of application work. CPU B, still executing OS/VS2 code, goes through the task dispatcher and turns control of this CPU from OS/VS2 to an application. Just prior to this, however, it issues a reset, then a restart signal to the other CPU to cause an interruption in CPU A and to cause CPU A to execute VS2. CPU A takes the signal interruption generated as a "farewell" by CPU B and starts executing OS/VS2 while CPU B processes independently. Thus OS/VS2 proceeds through the task dispatcher to turn control over to an application. At this point, both CPUs can be executing unrelated applications in parallel and in different virtual regions. It is also possible to have both CPUs executing code in OS/VS2, or a single multitasked region, at the same time. Processing continues in both CPUs until an interruption occurs.

Figure 3 Activities of OS/VS2 in MP Systems



*WHILE THIS DIAGRAM SHOWS SEPARATE ROUTINES BASED UPON CPU IDENTITY, VS2 MAKES USE OF COMMON ROUTINES FOR MANY OF THESE DECISIONS AND ANALYSES.

Interruptions occur in either CPU for exactly the same reasons as in uniprocessors. For example, a processor issuing a START I/O instruction receives the normal I/O interruption(s) signalling device-end, channel-end. The point, then, is not to get confused as to which CPU handles interruptions. After servicing an interruption and prior to relinquishing control of the CPU, VS2 executes its task-dispatching algorithm to determine which work to dispatch next. Thus, over a period of time, work will be passed back and forth between two CPUs because of interruptions and its changing status on OS/VS2's task-dispatching list. An MP system is considered *symmetrical* when both CPUs possess a like set of hardware features and channel paths to I/O devices. The converse of symmetry is *asymmetry* where only one CPU has a particular hardware facility (such as an emulator) or access to a needed device.

When necessary, CPU hardware will prevent one CPU from interfering with another in the midst of instruction execution through a process called interlocking which involves CPU hardware. A new software implementation called locking ensures against unforeseen altering or accessing of main storage at critical times by the other CPU.

The preceding discussion dealt with task dispatching and interruptions handled by the processors as the result of I/O activity. Like a uniprocessor, each MP processor has I/O devices attached to it through channels. The most flexible and desirable configuration is for the two processors to be symmetrical in every respect – that is, the two CPUs are identical in features and facilities and each CPU has a channel path to each control unit/device. In this way, OS/VS2, when dispatching tasks, is given maximum latitude as to which processor can (at any one instant) execute a particular task or initiate requested I/O operations. If a feature exists on only one CPU, OS/VS2 job control provides for assignment of a particular task to a specific CPU. Where I/O is the problem (that is, a system service request routine determines that it needs to access an asymmetric device), an affinity for a CPU is established only during the period when I/O is requested for the asymmetric device; otherwise, either CPU can execute the task's code.

This discussion of how work is managed and allocated by OS/VS2 in an MP configuration has assumed normal operating conditions. Such will not always be the case, however; thus, error recovery needs to be addressed. Significantly, within an MP system nearly every element is redundant or accessible through more than one path.

Specifically, at the CPU level, either processor can execute application or SCP work, and OS/VS2 provides for Alternate CPU Recovery when one CPU has failed. The OS/VS2 SCP or the console operator of OS/VS2 can vary a processor offline with the expectation that the remaining CPU will carry the whole load. An operator VARY command also permits other hardware components such as storage, control units, channels, and devices to be brought either offline or online in an orderly fashion. The next section discusses the extensive System/370 facilities in the Model 158/168 MP for handling error conditions and signaling malfunction notification within the system. The MP system with its single copy of the control program also assists in cases of device/channel error by providing such facilities as Alternate Path Retry (to another channel or control unit), Dynamic Device Reconfiguration (for mountable volumes on tape/DASD), a copy of the Link Pack Area Library, and the Clear I/O function (which clears channels or subchannels). Except for Alternate CPU Recovery and inter-CPU signaling, OS/VS2 also provides this error recovery to uniprocessors.

Each CPU in an MP configuration has an Interval Timer, CPU Timer, Clock Comparator, and Time-of-Day (TOD) Clock. What OS/VS2 provides, however, is centralized control of the Time-of-Day Clock in MP mode so that a task will get a single TOD clock image regardless of which CPU is executing it at any particular time. This is accomplished through a combination of

OS/VS2 code (which is executed at system IPL time or when VARYing a CPU online) and MP hardware called TOD *clock synchronization.*

The previous discussion attempted to establish the characteristics of System/370 MP by describing the process of work/task handling. With this environment defined, the purpose of the next sections is to discuss the specific multiprocessing hardware units.

Multiprocessing hardware

configuration

System/370 tightly-coupled multiprocessing is a configuration of the following components:

- Two Model 158 MP or two Model 168 MP CPUs with processor storage associated with each CPU.
- A unit connecting the processors which accomplishes the address selection and sharing of main storage and inter-CPU communication.
- A complement of channels associated with each CPU with most control units capable of being accessed by both CPUs via two-channel switching.
- A configuration control panel that orchestrates the above.

The differences in the details of the hardware between the Models 158 MP and 168 MP are delineated in Table 1. In simplest terms, the processor storage on each Model 158 must be identical in size whereas Model 168s can have any combination of storage sizes. When configured as uniprocessors, however, the CPUs can have individual increments of storage assigned to them. The actual size of the smallest assignable increment is 256K, 512K, or 1 megabyte depending on the total installed storage.

Another consideration is the real-addressing range of each storage box itself. Each physical storage increment does not come

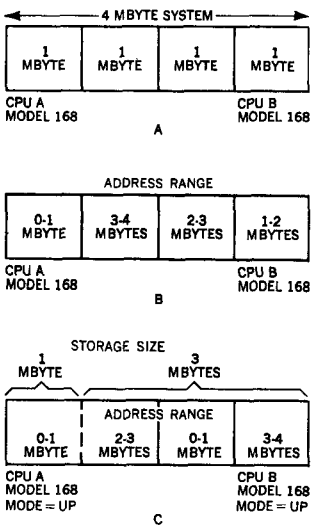
Table 1 Comparison of System/370 Model 158 MP and Model 168 MP

Function	*158 MP*	*168 MP*
Minimum storage	512K	1 megabyte
Maximum storage	4 megabytes	8 megabytes
Maximum storage for entire system	8 megabytes	16 megabytes
Identical storage size required for each CPU	Yes	No
Uniprocessor mode	Yes	Yes
Processor storage partitioned equally on uniprocessors	No	No
Processor storage assigned contiguously to CPU	No	No
Control over which CPU oscillator pulses the TOD clock	No	Yes

with preestablished address ranges as System/370 uniprocessors do. While this may seem quite different from storage on non-MP systems, all the specifics are resolved in a flexible manner at the configuration control panel of the MP system. At the panel the computer operator makes several determinations including:

- Mode of system operation—uniprocessing or multiprocessing.
- Which logical storage units are assigned to which processors if in uniprocessor mode.
- What the real address range will be on each unit of processor storage, regardless of mode.

Figure 4 MP storage addressing



For example, assume a four megabyte Model 168 MP system with logical storage units of one megabyte as shown in Figure 4A. Depicted in Figure 4B are example floating addresses. Assignments in uniprocessor mode could give CPU A one megabyte and CPU B three megabytes of processor storage and these could be addressed and partitioned, as illustrated in Figure 4C.

Similar flexibility is desired in assigning control units between CPUs and their channels. Dual (or more) paths to all devices, called *I/O symmetry*, enables either processor to undertake I/O, enhances availability in multiprocessing mode, and provides configuration flexibility from a single control panel when in uniprocessor mode.

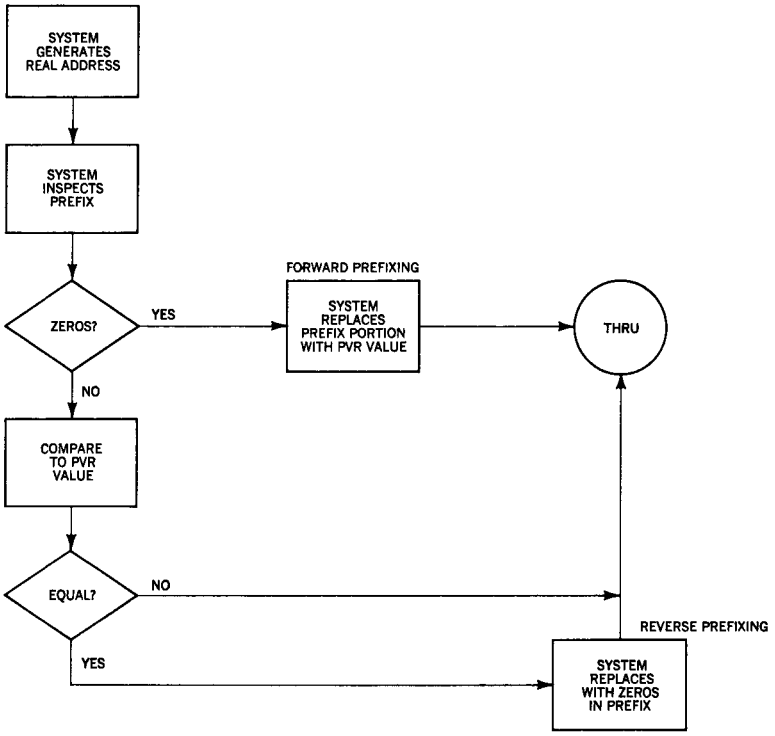
prefixing

Previous sections have explained that each CPU in a multiprocessing system has all the functions of a uniprocessor with regard to instruction execution, I/O operations, machine-check handling, and generating and servicing interruptions. However, interruptions not only involve CPU hardware but also utilize fixed processor-storage locations for storing PSWs, logout areas, and other machine-dependent information.

A uniprocessor utilizes main storage locations 0-4095 for these machine-dependent purposes. However, in multiprocessing mode, with main storage shared and available to both CPUs, complications arise. Since both processors in MP mode cannot use locations 0-4095 for the same purpose, another solution must be found. The following discussion, showing that neither CPU accesses real locations 0-4095 for normal usage, describes that solution.

The formal hardware term for the vehicle that solves this dilemma is prefixing. Prefixing on System/370 is completely configured by software (OS/VS2), is both forward and backward, and reserves actual real locations 0-4095 for OS/VS2 usage and crisis intervention.

Figure 5 Prefixing

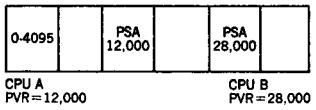


Each CPU in an MP configuration has a Storage Control Unit (SCU) within which is an internal hardware register called the Prefix Value Register (PVR). The PVR is effectively 12 bits long and is loaded by OS/VS2 using a new privileged instruction: Set Prefix (SPX). Store Prefix (STPX), another new instruction, stores the contents of the PVR for inspection. After the CPU develops a real main storage address (with or without using the DAT facility), prefixing hardware matches bits 8–19 (called the prefix portion) of the 24-bit real address that the CPU has developed against the contents of the PVR.

Three possiblities are defined: (1) If bits 8–19 (the prefix portion) of the 24-bit address are zero (that is, the address falls within the range 0-4095), these bits are replaced with the PVR contents and the prefixed address then falls within a 4096-byte, system-established area called the Prefix Storage Area (PSA). This case is called *forward prefixing*. (2) If prefix bits 8–19 of the 24-bit address are not zero and do not match the PVR, the 24-bit address is unchanged. The prefixing hardware then has established that the real address falls neither into the 0-4095 range nor into the PSA. (3) *Reverse prefixing* is the third possibility and occurs if bits 8–19 of the 24-bit address exactly match the PVR. (This address falls within the PSA.) Prefixing hardware, in such cases, alters the real address by replacing bits

8–19 of the address with zeros and thus forces an address in the real storage range 0-4095. Reverse prefixing is one way in which OS/VS2 can store hardware-related information or status into a unique real area (0-4095) that can be inspected by the other CPU when signalled. Machine-check handling is an example. The flow of this logic scheme is shown in Figure 5.

Figure 6 Prefix storage locations



In shared main-storage MP, OS/VS2 establishes two prefix storage locations (one PSA for each CPU), fixes their page frames, and loads bits 8–19 of their page-frame addresses into the PVR of each CPU. This is shown in Figure 6 using an illustrative real decimal address for PSAs of 12,000 for CPU A and 28,000 for CPU B.

Reverse prefixing occurs whenever OS/VS2 deliberately wishes to place information in real locations 0-4095; this is accomplished by using storage references within the range of the PSA. Also, during certain status-recording sequences, the CPU does not prefix, but places data directly into permanent lower storage. Such is the case during a Store Status operation, which results from a manual key being activated on the CPU console or from a Signal Processor order (discussed in the following section). All this is possible because, normally, real locations 0-4095 are not accessed by either CPU.

On multiprocessing configurations prefixing is always active regardless of CPU mode (UP or MP) or whether Translate is ON or OFF. When Translate is ON, prefixing is applied after translation. (Prefixing is not operative during the following operations: store status, channel references to extended logout area, I/O data transfer, accessing of data-addressing-words, and CCW fetching.)

contention

Advanced Function Extended embodies a number of hardware innovations for specifically resolving those areas of contention that can arise within the CPU.

Hardware serialization. In uniprocessor mode a CPU is neither aware of nor affected by other CPUs. Instruction execution and storage references take place in proper sequence without interference. With tightly-coupled multiprocessing configurations now possible within a virtual storage environment, System/370 architecture establishes a specific set of hardware rules that apply to proper instruction execution in such configurations. This set of rules results in serialization which ensures that activities occur in their proper sequence and thus guarantees the proper outcome of the various activities underway or about to start. A need for serialization becomes more evident since some System/370 models prefetch instructions and, in certain cases, affect storage accesses in a sequence different from instruction execution. These internal architecture characteristics are transparent to the

program and programmer since they are dictated by CPU logic hardware. However, they do impact and determine CPU performance characteristics, which vary among the System/370 models. Program results across all models, however, will be the same.

While hardware serialization is a basic part of System/370 uniprocessor architecture, tightly-coupled multiprocessing further establishes the rationale for serialization when one considers the need for another CPU to access the "serialized" CPU's main storage. Here the essential requirement is for the "serialized" CPU to accomplish its instruction execution without any uncertainty whether the accuracy of its results will be affected by another CPU's activities. Serialization ensures that main storage, high speed buffers, and storage-protection keys are not modified in an undetected fashion by another CPU (or its channels). Serialization within a CPU guarantees that another CPU or channel that observes its activities will do so in proper sequence.

Serialization occurs during hardware interruptions and while executing these instructions:

- Branch on Condition
- Compare and Swap
- Compare Double and Swap
- Store Clock
- Test and Set
- Load PSW
- Purge TLB
- Set Prefix
- I/O Instructions
- Read Direct
- Write Direct
- Signal Processor

Along with serialization in System/370, MP systems implement *interlocking* which prevents a CPU from observing or modifying a storage location being worked on by a Test and Set or Swapping instruction issued by the other CPU.

Locking. Locking is an OS/VS2 software mechanism and structure that establishes a sequence over system resources that cannot be shared or executed in parallel by multiprocessors. Their use is serial as controlled by locks that are "opened" and "closed" through the VS2 macro SETLOCK. To complete this structure, System/370 provides new hardware to facilitate lock setting/checking and to prevent contention during the critical lock-setting procedure. There are two new instructions, Compare and Swap, and Compare Double and Swap, and the former is used to accomplish this job. During execution of the swapping instructions, CPU operation is interlocked to eliminate the pos-

sibility of the word, which is being examined in main storage during instruction execution, being altered by the other CPU. This is in sharp contrast to the Or Immediate (OI) instruction which does not interlock storage and therefore provides no protection from the other CPU's activity. The swapping instructions function as follows (recall that each MP CPU has its own unique general purpose registers; they are not shared like main storage):

- Examine the contents of a general purpose register and a word in main storage.
- When the comparison is equal, the word in main storage is replaced by the contents of another register.
- When the comparison is unequal, the word in storage replaces the register contents originally used in the comparison.

The two varieties of swapping differ only in the amount of data compared (word or double word). They are most useful in handling parameter lists of addresses (because of use of words or double words) and switch testing/setting.

Disabling. Disablement describes the case in which a CPU masks out interruptions. During certain critical operations all interrupts including Malfunction Alert are masked out. (Disabled loops do not exist as in OS/MVT since an enabled window for Malfunction Alert and Emergency Signal is included in such loops in OS/VS2 Release 2.) These interrupts remain pending (and requested service is not performed) while the receiving CPU is preoccupied with the critical operation with interruptions masked. To reduce the occasion for disablement, VS2 utilizes Compare and Swap for updating of locations that might be simultaneously accessed by two CPUs without using locks, as shown by the following example:

```
        L 3,0(2)        Load general register 3 from storage
LOOP    L 4,3           Also load general register 4
          .
          .             Modify contents of general register 4
          .
          .
        CS 3,4, 0(2)    Store new value if storage location is un-
                        changed
        BNE LOOP
```

OS/VS2's structure of locks, the significant degree of parallelism possible in the control program, and System/370 hardware serialization all combine to allow more OS/VS2 code over a period to time to run with interruptions enabled (compared to the Model 65 MP and the MVT and OS/VS2 Release 1 software systems).

protection

There is no new protection hardware in System/370 MP configurations, except for the significant new system protection provided through a combination of some new instructions and new OS/VS2 architecture. First, the entire OS/VS2 SCP does not run under storage-protection key 0. Instead keys 0–7 are reserved for systems code and those areas such as Systems Queue Area (SQA) that can be accessed by other regions. This means that VS2 is reserving key 0 (and its prerogatives) to only a select portion of the system while extending normal storage protection groundrules to SCP code, thus protecting itself. Second, VS2 isolates itself and other users from "game playing" activities within a region by giving a user no ability to obtain control in key 0 or disrupt the system through improper addressing (deliberate or unintentional). VS2 responds to user service requests by operating under the storage protection key of the user rather than key 0. (The new PSW Key Handling instructions implement this and are discussed later in this paper.) Finally, OS/VS2 Release 2 provides multiple, virtual address spaces—one for each job, each TSO user, and each IMS application. This is a departure from OS/VS2 Release 1 with its single virtual address space for all regions and only two segment tables. What Release 2 provides then, with its multiple virtual address spaces, is a protection mechanism through the DAT facility that makes it impossible for a user to ever address outside his own address space.

Thus VS2 makes use of the hardware storage-protection key, reserving it for supervisor protection from user code, utilizing it when the control program seeks to isolate a user from others when performing systems services, and employing selective keys (9–15) to protect V = R code not subject to paging. (Key 8 is reserved for jobs executing V = V.) The reference and change-bit portion of the storage-protect key is as significant to VS2 as the access control code itself, relating not to protection but to management of the real main storage.

selective channel reset

Advanced Function Extended provides a new channel recovery capability supported only by OS/VS2 Release 2, to selectively reset a channel and is implemented by the Clear I/O (CLRIO) Function instruction on both the UP and MP. The Clear I/O Function accomplishes the resetting of a subchannel and the addressed device without affecting the status of other channels or subchannels.

The ability to selectively reset a subchannel by programming is significant and holds great promise for improved system operation by providing an added means by which an ailing I/O problem can be helped and a CPU problem avoided. Without the Clear I/O Function, prior systems were forced to re-IPL the entire system if the I/O path or device requiring reset was vital to system operation.

hardware resets

System/370 architecture, multiprocessing, and Advanced Function Extended hardware combine to provide a complex hierarchy of hardware resets. This section discusses the variously defined resets that can arise due to activation of the SYSTEM RESET and LOAD keys on the system console. Of significant importance is the fact that these hardware resets can also be initiated by means of program control. (This aspect is discussed later in this paper.)

A *CPU reset* clears check conditions, terminates instruction execution, clears pending interruptions, and invalidates the Translation Look-aside Buffer (TLB). A more informative way to view CPU reset (and realize that it preserves a large amount of CPU information) is to consider the function of an initial CPU reset.

An *initial CPU reset* performs the function of CPU reset and also zeros the current PSW, the prefix value register, the Clock Comparator, and the CPU Timer. Control registers are set to their initialized values.

A *program reset* causes a CPU reset and an I/O system reset in all channels tied to this CPU.

An *initial program reset* causes an initial CPU reset and an I/O system reset.

A *system clear reset* performs an initial CPU reset and an I/O system reset, zeros and validates main storage and its protection keys, validates and (depending upon the System/370 model) zeros the general and floating-point registers, and zeros the interval timer.

Normally the CPU enters the stopped state after each of these resets. The only exception is when a hardware-IPL sequence follows the reset that is triggered by activation of the LOAD key.

Summarized in Table 2 are the resets previously discussed. The ENABLE-SYSTEM-CLEAR key, referenced in the table, is on the system console. This summary relates to uniprocessors; in MP mode, however, the activation of SYSTEM RESET or LOAD keys also has an effect on the other CPU. Table 3 summarizes these resets.

inter-CPU communication

Multiprocessing architecture establishes a comprehensive procedure that enables one CPU to communicate with another in the normal course of operation in MP mode. The prime vehicle is a new instruction (discussed later) called Signal Processor (SIGP). There is also provision for response to crisis conditions. To simplify our discussion of this architecture, the two CPUs are referred to as either a *sender* (in that the CPU gen-

Table 2 Hardware resets

Key activated	*Setting of ENABLE-SYSTEM-CLEAR key*	*Reset*
SYSTEM RESET		
CPU with Store Status*	Normal	Initial program†
CPU with no Store Status*	Normal	Program†
	Clear	Systems clear†
LOAD	Normal	Initial program, then IPL
	Clear	System clear, then IPL

*A consideration only on a non-MP configuration.
†CPU then enters stopped state.

Table 3 Effect of SYSTEM RESET or LOAD keys

Key activated on CPU console	*Setting of ENABLE-SYSTEM-CLEAR key*	*Reset propogated to other CPU*
SYSTEM RESET	Normal	Program reset
	Clear	System clear
LOAD	Normal	Program reset
	Clear	System clear

erates a signal and expects, perhaps, a response) or a *receiver* (the recipient of the sender's signal). There are cases, however, when a CPU can signal itself, if for no other reason than to identify itself to VS2 or to obtain status information.

Inter-CPU communication at the receiving CPU appears as an external interruption with these characteristics: bit 6 of the interruption code is set to 1 and, in permanent lower storage locations 132–133, the CPU address of the sender is set. A variety of inter-CPU interruptions is now given.

The *Malfunction Alert* interruption is automatically generated by the alternate CPU to show that it is entering the check-stop state or has lost power. This can be masked out by bit 16 of control register 0.

Emergency Signal is not hardware-generated as is Malfunction Alert; it results from a Signal Processor instruction (SIGP) issued by the sender. Control register 0, bit 17 provides masking in the receiver.

External call results from a SIGP instruction issued by the sender. Masking is by control register 0, bit 18.

The *Time-of-Day (TOD) Clock Synchronization check* interruption occurs when two TOD clocks are running but get "out of synch" (that is, bits 32–63 do not match). This can be masked out by control register 0, bit 19.

signal processor

The receiving CPU is also subject to control from the sending CPU as the result of orders sent by the SIGP instruction. Specifically, the sender can request from the receiver:

- SENSE—Deliver status information back to the requestor.
- START—Enter operating state, if stopped.
- STOP—Enter stop state.
- STOP and STORE STATUS—The stop state is entered and CPU status is saved in real storage locations (not subject to prefixing) 216–512.
- INITIAL MICROPROGRAM LOAD (IMPL)—The receiver performs initial program reset, then undertakes the IMPL sequence by reading the console file.
- RESETS—The manually-initiated resets (previously discussed) can also be signalled by SIGP to another processor (specifically—program reset, initial program reset, CPU reset, and initial CPU reset).
- RESTART—The receiver-CPU stores the old PSW in location 8, fetches a new PSW from location 0, and commences program execution. Restart is issued, for instance, during start-up of the system or when a CPU (the one being restarted) is being VARYed online.

  The sending CPU, the issuer of the SIGP instruction, is ordering the addressed CPU to do something, is signaling, or is requesting information. However, the effect and immediacy of the SIGP instruction on the receiver is a function of current activity in the receiver, its operating status, masking condition, and status of any prior SIGP orders that may be outstanding. When SIGP is executed on the sender:

- The condition code is set on the sender, indicating whether the order was accepted by the intercommunication hardware.
- With condition code = 0, the order is accepted and no status is stored.
- A condition code = 1 indicates the order was not accepted and status bits have been stored in a register to indicate why. Common reasons for rejection include an already-pending order that has not yet been completed in the addressed (receiving) CPU, manual intervention in progress there, or a condition in which the addressed CPU is check-stopped.
- Nothing occurs in the addressed CPU beyond returning status to the sender in the form of the status bits when SIGP requests sense information. The sense order is an example of a situation for which a processor might address itself in the

SIGP instruction. For instance, if a CPU has masked out external signals from the other CPU (all except restart can be masked), a SIGP issued to itself will return status information on pending external call and emergency signal interruptions.

An additional intercommunication consideration in System/370 MP systems is the role of buffer storage and the storage-protection keys found in each processor. The CPU hardware undertakes the following. In MP mode, a Set Storage Key instruction executed in one CPU results in setting the key in the protection array in both CPUs. Reference and change bits are also set in both arrays. (The reader is reminded that the physical storage protection array logically associated with each 2 K block of main storage may be housed separately from storage within the processor.) These actions are transparent to programming and are a function of internal MP architecture aimed at presenting and preserving the single-system image to the user.

timing facilities

The timing facilities available on the Model 158 MP and Model 168 MP configurations are the same as on the uniprocessors. What differs is the additional hardware provided in MP mode to synchronize the Time-of-Day (TOD) clock in each CPU. The various clocks on each processor are handled by OS/VS2 and provide the programmer with exactly the same services as in uniprocessing. Note, however, that a task is not necessarily executed on only one CPU since it will be passed back and forth as interruptions occur and dispatching is undertaken. Thus the application program and VS2 must have a consistent TOD clock reading if the presence of two CPUs (with two TOD clocks) is to be transparent and have no effect upon program results. Also, on each CPU the Clock Comparator is matched against its TOD clock during its operation.

TOD clock synchronization is the hardware solution to this dilemma and requirement. Sychronization applies only in MP mode and is controlled by a bit setting in a control register. When the Set Clock instruction is executed, a value provided by the operator is loaded into the TOD clock. Clock operation following Set Clock now depends upon control register 0. When bit $2 = 0$ (its initial value and the value used in the uniprocessor mode), the TOD clock enters set state and starts running immediately following Set Clock. However, when the TOD synch bit = 1, the clock enters the stopped state until either of two events occur:

- TOD synch bit = 0.
- The other CPU's running TOD clock is incremented to 0s in positions 32–63.

Table 4 Layout of control register 0

Bit	*Function*
1	Set System Mask suppression
2	TOD synchronization control
8–9	Page size control 01 = 2K bytes page size 10 = 4 K bytes page size
11–12	Segment size control 00 = 64K bytes segment size
16	Malfunction Alert mask
17	Emergency Signal mask
18	External Call mask
19	TOD Clock Synchronization Check mask
20	Clock Comparator mask
21	CPU Timer mask

Table 5 Programming support that utilizes the new instructions

	OS/VS2 Release 2 and above	
VTAM	*Uniprocessors*	*MP mode*
Swapping	Swapping PSW key handling Clear I/O	Signal processor Set/Store prefix Store CPU address

Either event is under the control of OS/VS2 and occurs when VS2 has completed its time-setting sequence.

The TOD clocks in the two multiprocessing CPUs are pulsed by separate oscillators when operating in UP mode. In MP mode, a single oscillator pulse is used to increment both TOD clocks. On the Model 158 MP this is always the oscillator on CPU A; a switch is provided on the Model 168 MP configuration panel to allow the operator to select the oscillator to be used. On each CPU, the CPU timer and clock comparator are stepped in the same increments as that CPU's TOD clock. This is true in both UP and MP modes of operation.

Set-clock can be issued by VS2 on either CPU provided at least one of the clock security switches is activated on ENABLE SET. During MP operation, if VS2 discovers one of the TOD clocks is malfunctioning or damaged, it will perform TOD services for the entire system using the remaining valid TOD clock.

control register usage

Only control register 0 is affected by and used for multiprocessing implementation. Table 4 depicts the layout of this control register as it pertains to advanced function (virtual storage addressing) or multiprocessing.

Table 6 Use of new instructions in System/370 Models

Instruction	*Model 115 and 125*	*Model 135*	*Model 145*	*Model 155-II, 158, 168, 158/168 MP in UP mode*	*Model 158/168 MP*
Swapping	Standard	Standard*	Standard*†	Standard	Standard
PSW key handling	No	No	Yes†	Standard	Standard
Clear I/O	No	No	Yes†	Standard	Standard
Signal Processor	No	No	No‡	No‡	Yes
Prefixing	No	No	No‡	No‡	Yes
Store CPU Address	No	No	No‡	No‡	Yes

*Provided as conditional swapping features.
†Provided on Model 145 through advanced control program support feature required by OS/VS2 Release 2 and above. Includes conditional swapping instructions.
‡May be issued by VS2 when changing from MP to partitioned mode, or from partitioned to MP mode or to detect presence or absence of multiprocessing hardware.

MP instructions and facilities

Advanced Function Extended introduces a series of new instructions that are available on specific System/370 models for possible use by a variety of systems control programming.[2] The instructions are listed below:

- Compare and Swap
- Compare Double and Swap
- Insert PSW Key
- Set PSW Key from Address
- Clear I/O
- Set Prefix
- Store Prefix
- Signal Processor
- Store CPU Address

Tables 5 and 6 associate these instructions with particular CPU models and the software that utilizes them. The remainder of this section is concerned with a discussion of the new instructions with emphasis upon their rationale and possible use.

Compare and Swap (CS). The execution of CS is serialized and the contents of a register (R_1) are compared with the word in the addressed main storage location: **CS**

CS R_1, R_3, D_2 (B_2)

One of two condition codes results:

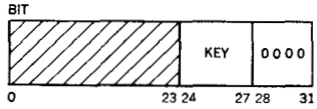
- 0—An equal comparison; the word in main storage is replaced by the contents of register R_3.
- 1—Unequal comparison; register R_1 is loaded with a word from main storage.

CDS *Compare Double and Swap (CDS).* This instruction operates in the same way as the Compare-and-Swap instruction, except that the comparison is a double word (eight bytes) in width:

CDS R_1, R_3, D_2 (B_2)

Swapping instructions are not privileged instructions and are most useful for altering the contents of main storage after the issuer checks to be sure that he knows what is supposed to be altered (an equal). When the comparison shows a descrepancy (unequal match), the information actually in main storage is copied into a register and thus presented to the user. The four-byte width of CS is useful for manipulating address lists while the CDS usage of a double word provides for both an address and a parameter entry number. The latter is helpful in ensuring that the sequence of a list is unaltered by another CPU.

Figure 7 Protection-key portion of current PSW as used by IPK



IPK *Insert PSW Key (IPK).* The protection-key portion of the current PSW is loaded into general-purpose register 2 as shown in Figure 7. This instruction is helpful to VS2, which can no longer assume that nucleus or other SCP code operates under key 0 and must therefore determine what the active key is at any point.

SPKA *Set PSW Key from Address (SPKA).* This instruction results in setting the protection key of the current PSW from bits 24–27 of the second operand:

SPKA D_2 (B_2)

This instruction is helpful in facilitating VS2's servicing of an application's private address-space request using the key of that job rather than the key of VS2. Thus VS2 protects itself and isolates others from the consequences of an unauthorized access. For instance, a simplified example might involve moving a logical record to a user-specified work area:

```
IPK                 VS2 key saved in general purpose register 2
L       3, USERKEY  Put user key in general purpose register 3
SPKA    3(0)        Key into PSW
MVC     ———         Move to work area
SPKA    2(0)        Restore VS2's key
```

CLRIO *Clear I/O (CLRIO) Function.* This instruction resets the subchannel and causes the current operation at the addressed device to be discontinued:

CLRIO D_2 (B_2)

SIGP *Signal Processor (SIGP).* An inter-CPU signal or request is generated in the following manner. R_3 contains the addressed CPU, D_2 (B_2) is the resultant address of an eight-bit order code sent to the addressed CPU, and R_1 contains the status as a result of SIGP's execution when the condition code = 1 (Such is the

case as a result of SIGP SENSE, or due to SIGP's order being rejected. If condition code = 0, 2, or 3 there are no R_3 status bits in R_1.)

SIGP R_1, R_3, D_2 (B_1)

One of the capabilities of the SIGP multiprocessing instruction is to request a new MP facility—Store Status. The SIGP order Stop and Store Status places the following information within the first 512 lower permanent locations of real main storage:

- CPU timer
- Clock comparator
- Current PSW
- Prefix value
- Model-dependent feature
- Floating point registers (0–6)
- General purpose registers (0–15)
- Control registers (0–15)

Store Status is not subject to prefixing and is a good example of the use to which VS2 puts real locations 0-4095 for common, systems reliability. The console operator can initiate Store Status manually by activating a key on the CPU console panel.

SPX

Set Prefix (SPX). The CPU's prefix register is set from bits 8–19 of the addressed word in main storage:

SPX D_2 (B_2)

This gives VS2 the flexibility of selecting any page frame in main storage as the PSA and loading the prefix portion of its frame address into the PVR.

STPX

Store Prefix (STPX). The contents of the prefix register are stored into the specified word (the format of which is depicted in Figure 8):

STPX D_2 (B_2)

STAP

Store CPU Address (STAP). The CPU stores into the designated half-word its own established address (X'0000' or X'0001'):

STAP D_2 (B_2)

Considering that VS2 can be executing on either or both CPUS at a given point in time, the STAP instruction conveniently provides the means for determining which CPU it is currently controlling. (The Store CPU Address instruction should not be confused with the System/370 instruction Store CPU ID, which is not associated with multiprocessing.)

Figure 8 Contents of the prefix register

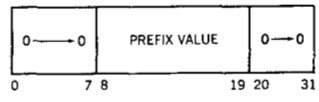


Table 7 Comparison of multiprocessors

Features	*System/370*	*Model 67 Duplex*	*Model 65 MP*
Two CPUs sharing main storage	Yes	Yes	Yes
Use of virtual storage	OS/VS2	Yes	No
Multiple virtual storage	Yes	Yes (TSS)	No
Configuration control panel			
MP or UP mode	Yes	Yes	Yes
Floating storage	Yes	Yes	Yes
Channel and I/O switching	Yes	Yes	Yes
Interval timer	Yes	High resolution	Yes
CPU timer	Yes	No	No
Clock comparator	Yes	No	No
TOD clock	Yes	No	No
Hardware synchronization clocks	TOD	No	No
Malfunction signal	Yes	Yes	Yes
Inter-CPU signalling	SIGP	Extended direct control	Direct control

Comparison of multiprocessing systems

System/370 158/168 MP hardware has evolved from earlier IBM Systems—System/360 Model 67 Duplex and Model 65 MP. Several features of these prior systems are presented next and placed in the perspective of System/370 followed by discussions of each system. A recapitulation of the multiprocessing hardware features of the three systems is contained in Table 7. A fourth multiprocessor, the IBM 9020, used at the Federal Aviation Administration, has been described elsewhere.[3]

features of prior systems

Special purpose. The System/360 Model 67 Duplex and Model 65 MP were unique systems in that they employed specially developed hardware beyond System/360 architecture, and they required programming support other than operating systems available at that time (OS/360, DOS/360). The Model 67 Duplex is supported by the Time-Sharing System/360 (TSS) and by user-modified code based upon CP-67. Several timesharing services still use the Model 67 Duplex and their version of virtual machines. The Model 65 MP was implemented using special versions of OS/MVT. In contrast, System/370 Models 158/168 MP are members of System/370 family and utilize standard OS/VS2 support with Release 2. In particular, OS/VS2 is designated as Systems Control Programming (SCP) and this operating system also supports uniprocessor configurations. Thus MP is not grafted upon a reluctant operating system environment, but is rather an integral part of the whole system.

System response to malfunction. System/370 multiprocessing, with its large main storage and multiple virtual address spaces, places a large number of users within a single data processing

environment. Because of this, the System/370 MP configurations were designed with an added capability to handle a system malfunction. For instance, the Signal Processor (SIGP) instruction provides a greatly enhanced inter-CPU communication facility that facilitates recovery subsequent to CPU failures. The new System/370 Clear I/O (CLRIO) Function instruction, offers promise of avoiding the occurrence of a channel or channel control unit/device problem necessitating the IPL of a CPU. Also, the dual paths now available on the IBM 3705 programmable communications controller running under Network Control Program/VS remove the prior System/360 two-channel restrictions to the IBM 2701/2702/2703 (which meant that the manually-IPLed CPU acquired teleprocessing affinity).

Storage manangement. System/370 MP employs virtual storage management under OS/VS2 control. In addition, multiple virtual address spaces are implemented—one for each user. The Model 67 Duplex under TSS also employs virtual storage through the DAT feature and provides multiple virtual storage. The Model 65 MP, of course, made no use of virtual storage or DAT. All systems employed prefixing, but in different ways.

Configuration changes. System/370, the Model 67 Duplex, and the Model 65 MP share the objectives of shared main-storage, pooled I/O devices through multiple data paths (channels), and the ability of each CPU to be operated separately in uniprocessor mode. In all these systems, a hardware configuration panel is provided to facilitate operator-initiated configuration and reconfiguration of the CPUs, storage, and/or I/O devices (although each system employs different implementations to accomplish this).

Inter-CPU communication. All three MP systems under discussion have methods by which the CPUs communicate with one another. Each provides for signaling the other CPU in case of machine check and to exchange information. Both the Model 67 and the Model 65 MP accomplish signaling between CPUs by modification of the Direct Control hardware. On the other hand, System/370 MP architecture contains a specific instruction for communication—Signal Processor, which is more comprehensive in its function and in its effect on multiprocessing operation than its predecessor mechanisms. Through the use of this instruction, a functioning CPU can, for example, start, stop, or reset the other CPU, or cause the other CPU to perform a logout operation. These capabilities enhance the prospects for system recovery in the event of a CPU failure.

Model 67 duplex

The Model 67 Duplex, like System/370, implements shared main storage multiprocessing with virtual storage. The programming support is provided through TSS while other users have

developed their own programming support based upon CP-67 with its virtual machines.

There are no special hardware instructions on Model 67 Duplex for multiprocessing purposes. Forward prefixing is employed on the Model 67, with the specific assignment of the alternate address and user selection performed through hardware manually set by a customer engineer. The non-alternate area, unlike System/370, occupies real storage locations 0-4095.

A configuration control panel gives the machine operator the ability to determine systems mode (simplex/uniprocessing or duplex/multiprocessing), configure main storage units through floating address switches, and, through a series of switches, configure channels and control units. What is unique about the Model 67 is that each CPU can sense and store into its control registers (8–14) the manual switch settings on the configuration unit by use of the Store Multiple Control instruction.

Inter-CPU signaling utilizes the Model 67 Extended Direct Control facility and a control register. Each CPU can receive from the other an external interruption caused by a malfunction alert generated by a machine check or a Write Direct instruction. (Write Direct is the vehicle by which one CPU signals the other. It also provides a means for an "external start" by which the sending CPU can terminate operations in the receiving CPU and cause a PSW-restart from location 0.) The Model 67, unlike System/370, does not provide in its signaling design the extensive control, resets, sense, and status capability provided by the System/370 SIGP instruction. Also, the Model 67 does not have a Clear I/O Function instruction for selective resetting of subchannels.

Timing facilities on the Model 67 are equivalent to System/360 with the exception that only a 13-microsecond high-resolution timer is provided on each CPU. There is no special hardware on the Model 67 to synchronize timing facilities between the CPUs; this is handled entirely by programming.[4]

To resolve contention and to ensure that two CPUs are not interfering with one another, a duplex System/360 Model 67 must, by necessity, operate in a disabled mode for a certain amount of time. In this case, one CPU must mask out all interruptions including, possibly, those from the other CPU. Also, a certain amount of time is spent looping on program switches while one CPU awaits the freeing up of a resource by the other. In System/370 multiprocessing, disablement and looping still occur, but VS2 architecture and the hardware mitigate the consequences by directly addressing both of these multiprocessing bottlenecks

with a more sophisticated structure of locks, new instructions, and hardware synchronization.

Model 65 MP

The Model 65 MP differs from Model 67 and System/370 in its management of real storage, in that there is no virtual storage or DAT facility. The Model 65 MP program support consists of modification to OS/MVT. Like the Model 67 and System/370, the Model 65 MP does share real main storage between two CPUs, but like the Model 67 there are no special CPU instructions that support this environment. Forward prefixing is used and is wired manually like the Model 67 but, unlike the Model 67, the Model 65's prefixing hardware must address the first 4K and last 4K bytes of available main storage as prefix storage areas. There is no lattitude as to the location of PSAs beyond the operator selection of which CPU will utilize which of the two areas. Additionally, unlike System/370, the Model 65 utilizes real locations 0-4095 for normal systems operations such as PSW storing.

A configuration control panel provides for storage allocation by floating address of the main storage units, I/O control unit allocation, mode of operation, and prefix area selection. The Model 65 MP, however, does not have control registers, and there is no ability for the CPU to sense these manual switch settings as with the Model 67.

Inter-CPU communication, similar to the Model 67, is accomplished either through receipt of Malfunction Alert generated by the other CPU's machine check or through an external interruption signal generated by the sender with the Write Direct instruction of the Direct Control Feature.

The previous comments relating to timing facilities, contention, and locks for Model 67 also apply to the Model 65 MP, with the exception that the 13-microsecond timer is not available on the MP models of the Model 65.

Summary comment

System/370 implements multiprocessing by standard options of OS/VS2 at Release 2 and above. Available real storage increments are much larger than those available on the Model 67 or Model 65 MP; monolithic storage rather than core storage technology is utilized. Virtual storage with multiple virtual address spaces is inherent to System/370 MP systems.

Several special instructions are provided on System/370—some for implementation on uniprocessor configurations under OS/VS2, others specifically for MP support. The PSW key han-

dling instructions provide for a more secure control program; the swapping instructions facilitate a locking structure less likely to "shut out" the other CPU unnecessarily; processor signaling provides a very complete array of inter-CPU communication, resetting, and "pulse-taking". These instructions are designed either specifically for multiprocessing or in anticipation of a multiprocessing environment. Experience with the System/360 Model 67 and the Model 65 MP has led to the requirement for and implementation of the SIGP facility. Likewise, System/370's Clear I/O Function instruction profits from past experience in which a channel malfunction could prove fatal to the entire system even if for no other reason than the requirement of a re-IPL of the entire system to reset the offending hardware.

Prefixing on System/370 is improved in its implementation. First, it is completely under program, rather than operator, control and OS/VS2 has complete freedom in where it defines prefix save areas. Also, each CPU's storage control unit contains a prefix register that is loaded and inspected under program control. Thus the lower absolute 4096 bytes of real storage are not used for prefixing, which means that on certain machine-initiated conditions, activation of Store Status facility, or by request of VS2, the lower 4096 bytes of real main storage are available to capture critical information without disturbing either CPU's prefix save area.

The System/370 configuration control panel is similar to that of the Model 65 MP and Model 67, but unlike the Model 67 it does not provide any facility for CPU sensing of switch settings. There are no prefixing switches because prefixing is program controlled (rather than "hard-wired"). The System/370 configuration panel unlike the Model 65 MP or Model 67, is microprogrammed to detect a number of invalid configuration settings and thus to prevent these configurations from being entered. The System/370 panel, unlike the Model 65 MP, is not a "hot" panel; that is, configuration changes do not occur when a dial or switch is changed, but rather take effect only when ENTER CONFIGURATION is pressed.

System/370 inter-CPU communication is accomplished using the Signal Processor instruction. Nothing comparable, either in scope or in basic architecture, is provided in MP predecessors. In addition, System/370 provides the instruction Store CPU Identification, implements hardware serialization of functions, hardware interlocking of storage, and provides OS/VS2 with new instructions to assist with its implementation of a multiprocessing system.

System/370 has, as part of its multiprocessing hardware, a facility for coordinating the operation of the Time-of-Day Clock on

each CPU. Its function is to ensure that both CPUs have TOD Clocks with identical readings. Since task execution, generally, can "flip-flop" between CPUs, the need is evident for a single TOD appearance. The Model 67 and Model 65 MP derive time-of-day from an interval timer without the benefit of hardware-synchronized timers or clocks.

Concluding remarks

This paper has described hardware extensions to System/370 to provide multiprocessing. While the emphasis is upon tightly-coupled implementation through MP hardware and OS/VS2 Release 2, the characteristics of loosely-coupled systems have been sketched out for contrast. The comparison of IBM multiprocessors has been included to provide a rationale for the evolution of System/370 multiprocessing architecture with significant differences from and improvements upon prior System/360 implementation.

ACKNOWLEDGMENT

The author benefited from the creative suggestions and support from several individuals at the IBM New England Systems Support Center.

CITED REFERENCES

1. J.S. Arnold, D.P. Casey, and R.H. McKinstry, "Design of tightly-coupled multiprocessing programming," in this issue.
2. *IBM System/370 Principles of Operation*, Form No. GA22-7000, International Business Machines Corporation, Data Processing Division, White Plains, New York.
3. "An application-oriented multiprocessing system," *IBM Systems Journal* **6**, No. 2, 78–132 (1967).
4. *IBM System/360 Model 67 Functional Characteristics*, Form no. GA27-2719, International Business Machines Corporation, Data Processing Division, White Plains, New York.

GENERAL REFERENCE

A.L. Scherr, "Functional structure of IBM virtual storage operating systems Part II: OS/VS2-2 concepts and philosophies," *IBM Systems Journal*, **12**, No. 4, 382–400, (1973).