3.0—Multiprocessing

Terms you’ll need to understand:

✓ Multiprocessing
✓ Internal register
✓ Network operating system (NOS)
✓ Loose coupling
✓ Tight coupling
✓ High-level cache
✓ Shared memory
✓ Symmetrical multiprocessing (SMP)
✓ Asymmetrical multiprocessing (ASMP)
✓ Scalable system architecture
✓ Coherent system
✓ Spinlock mode
✓ Microkernel architecture
✓ Process programming
✓ Multithreading
✓ Bus snooping
✓ Massively parallel processing (MPP)
✓ Overhead
✓ Latency
✓ Determinism
✓ Skew
✓ Steppings
✓ Bit ratings

Techniques you’ll need to master:

✓ Increasing the overall processing power of a single node
✓ Allocating resources
✓ Increasing the overall processing power of a system
✓ Differentiating between level 2 and level 3 cache
✓ Using multiple threads
✓ Flushing the cache
✓ Configuring network operating systems
✓ Preventing the system bus from becoming a bottleneck
✓ Determining the bit rating of a specified processor
✓ Tuning a server network’s performance
Introduction

Server+ Exam Objective 1.12 states that the test taker should know the features, advantages, and disadvantages of multiprocessing. This information includes knowledge about:

- Unix
- OS/2
- Windows NT/2000/XP
- Program kernel
- Xeon
- Pentium
- Node
- Embedded space
- Linux
- System bus
- Feedback loop
- Digital content creation (DCC)
- Windows XP Professional x64 Edition
- Windows on Windows 64 (WOW64) x86

Multiprocessing Systems

A computer’s capability to process more than one task simultaneously is called multiprocessing. A multiprocessing operating system is capable of running many programs simultaneously, and most modern network operating systems (NOSs) support multiprocessing. These operating systems include Windows NT, 2000, XP, and Unix.

Although Unix is one of the most widely used multiprocessing systems, there are others. For many years, OS/2 has been the choice for high-end workstations. OS/2 has been a standard operating system for businesses running complex computer programs from IBM. It is a powerful system, employs a nice graphical interface, and can also run programs written for DOS and Windows. However, OS/2 never really caught on for PCs.
The main reason why multiprocessing is more complicated than single-processing is that their operating systems are responsible for allocating resources to competing processes in a controlled environment.

With the growth of commercial networks, the practice of using multiple processors in embedded motherboard designs has become almost universal. Not too long ago, clients or network administrators constructed most multiprocessing configurations at the board or system level themselves. Today, motherboards are available incorporating multiple microprocessors on the same die.

A multiprocessing system uses more than one processor to process any given workload, increasing the performance of a system's application environment beyond that of a single processor's capability. This permits tuning of the server network's performance, to yield the required functionality. As described in Chapter 2, “Server Availability,” this feature is known as scalability, and is the most important aspect of multiprocessing system architectures. Scalable system architecture allows network administrators to tune a server network's performance based on the number of processing nodes required.

Collections of processors arranged in a loosely coupled configuration and interacting with each other over a communication channel have been the most common multiprocessor architecture.

This communication channel might not necessarily consist of a conventional serial or parallel arrangement. Instead, it can be composed of shared memory, used by processors on the same board, or even over a backplane. These interacting processors operate as independent nodes, using their own memory subsystems.

Recently, the embedded server board space has been arranged to accommodate tightly coupled processors, either as a pair, or as an array. These processors share a common bus and addressable memory space. A switch connects them, and interprocessor communications is accomplished through message passing. In the overall system configuration, the processors operate as a single node, and appear as a single processing element. Additional loosely coupled processing nodes increase the overall processing power of the system. When more tightly coupled processors are added, the overall processing power of a single node increases.

These processors have undergone many stages of refinement over the years. For example, the Xeon processors were designed for either network servers or high-end workstations. Similarly, Pentium 4 microprocessors were intended solely for desktop deployment, although Xeon chips had also been called “Pentiums” to denote their family ancestry. Pentium and Xeon processors are
currently named separately. The Xeon family consists of two main branches: the Xeon dual-processor (DP) chip, and the Xeon multiprocessor (MP).

**Multiprocessing Features**

Dual-processor systems are designed for use exclusively with dual-processor motherboards, fitted with either one or two sockets. Multiprocessor systems usually have room on the board for four or more processors, although no minimum requirement exists. Xeon MPs are not designed for dual-processor environments due to specific features of their architecture, and as such, are more expensive.

Dual processors were developed to function at higher clock speeds than multiprocessors, making them more efficient at handling high-speed mathematical computations. Multiprocessors are designed to work together in handling large databases and business transactions. When several multiprocessors are working as a group, even at slower clock speeds, they outperform their DP cousins. Although the NOS is capable of running multiprocessor systems using Symmetrical Multi-Processing (SMP), it must be configured to do so. Simply adding another processor to the motherboard without properly configuring the NOS may result in the system ignoring the additional processor altogether.

**Types of MPs**

Various categories of multiprocessing systems can be identified. They include

- Shared nothing (pure cluster)
- Shared disks
- Shared memory cluster (SMC)
- Shared memory

In *shared nothing* MP systems, each processor is a complete standalone machine, running its own copy of the OS. The processors do not share memory, caches, or disks, but are interconnected loosely, through a LAN. Although such systems enjoy the advantages of good scalability and high availability, they have the disadvantage of using an uncommon message-passing programming model.

*Shared disk* MP system processors also have their own memory and cache, but they do run in parallel and can share disks. They are loosely coupled through a LAN, with each one running a copy of the OS. Again, communication between processors is done through message passing. The advantages
of shared disks are that disk data is addressable and coherent, whereas high availability is more easily obtained than with shared-memory systems. The disadvantage is that only limited scalability is possible, due to physical and logical access bottlenecks to shared data.

In a shared memory cluster (SMC), all processors have their own memory, disks, and I/O resources, while each processor runs a copy of the OS. However, the processors are tightly coupled through a switch, and communications between the processors are accomplished through the shared memory.

In a strictly shared memory arrangement, all of the processors are tightly coupled through a high-speed bus (or switch) on the same motherboard. They share the same global memory, disks, and I/O devices. Because the OS is designed to exploit this architecture, only one copy runs across all of the processors, making this a multithreaded memory configuration.

**Importance of SMP**

The explosion of bandwidth for networking servers has placed unreasonable demands on single-processor systems, which cannot handle the workload! It must be distributed across multiple processors, using SMP. The main differences between an SMP system and every other system are that it utilizes more than one processor, and operates from a somewhat specialized motherboard. The main advantages to using SMP have to do with network speed and expense. Surprisingly, not only is an SMP solution faster than a single-processor solution, but also less expensive. Its higher performance is due to the fact that a multiprocessor motherboard can deliver multiple paths of data processing, whereas a single-processor motherboard can only harness the single processor's capabilities. Compare it to moving 20 students to a server networking class using either a turbo-charged Ferrari or four Volkswagen Beetles. Even though the Beetles are cheaper, they will also get the job done faster. Of course, driving to class in a Ferrari would be more fun!

Hardware performance can be improved easily and inexpensively by placing more than one processor on the motherboard. One approach is called asymmetrical multiprocessing (ASMP), where specific jobs are assigned to specific processors. Doing ASMP effectively requires specialized knowledge about the tasks the computer should do, which is unavailable in a general-purpose operating system such as Linux.

The second approach is the one mentioned often in this book, called symmetrical multiprocessing (SMP), where all of the processors run in parallel, doing the same job. SMP is a specific implementation of multiprocessing whereby multiple CPUs share the same board, memory, peripherals, resources, and operating system (OS), physically connected via a common high-speed bus.
Compared to ASMP, SMP is relatively easy to implement because a single copy of the operating system is in charge of all the processors.

In early SMP environments, programmers had to remain aware that because the processors shared the same resources, including memory, it was possible for program code running in one CPU to affect the memory being used by another. Programming for these types of situations required special protection. Often, process programming was normally only run on one processor at a time, keeping the process safe from intrusion. However, the program kernel was still subject to call by various codes running on different processors. One solution involved running the kernel in spinlock mode, where only one CPU at a time was serviced. Other processors seeking entry had to wait until the first CPU was finished. Although the system was protected from competing processors, it operated inefficiently.

SMP systems do not usually exceed 16 processors, although newer machines released by Unix vendors support up to 64. Modern SMP software permits several CPUs to access the kernel simultaneously. Threaded processes can run on several CPUs at once, yet without suffering from kernel intrusion. Recall that a thread is a section of programming that has been time-sliced by the NOS in order to run simultaneously with other threads being executed in an SMP operation.

Know that the main idea behind symmetrical multiprocessing is the use of multiple threads.

The benefits of properly using SMP server networking power are

- **Multithreading**—Unless communicating with another process, single-threaded programs do not gain from SMP.

- **Instruction safety**—Programs do not rely on unsafe non-SMP instructions, or improper thread priorities.

- **Processor utilization**—Subtasks are divided so that all CPU-bound instructions take no longer than 30 milliseconds.

**Symmetric Multiprocessing Environments**

Symmetric multiprocessing environments require that each CPU in the system have access to the same physical memory using the same system bus. Otherwise, the execution of a program cannot be switched between processors. All CPUs sharing the same physical memory must also share a single
logical image of that memory to achieve a *coherent* system. In a coherent system, all of the SMP CPUs see and read the identical data byte from the same memory address. SMP does not permit running of a unique copy of the OS on each CPU, nor does it permit each CPU to build its own unique logical memory image. A completely distinct image for each CPU would be chaos! Figure 3.1 depicts a simple SMP system.

![Figure 3.1](image-url) A simple SMP environment.

Four boxes represent four individual processors, each with its own on-chip level 1 cache. Data is transferred between a processor’s level 1 cache to a separate level 2 cache that is assigned to that processor. Data can be transferred to and from the L2 cache and real memory, or I/O devices for processing.

Memory coherence and shared memory are a great challenge to designers of multiprocessing architectures, further complicated by the existence of on-chip, high-speed cache memory. This high-speed cache is used to store values read by the CPU from system-memory locations. In a system that does not utilize high-speed cache, the processor repeatedly reads these memory locations during program execution. High-speed cache relieves the CPU from having to read this memory using the slower system bus. This reduces the drain on the system bus, and improves performance.

More processing time is saved during write caching because of the thousands of times that cache values might change before the processor either flushes the cache or writes its contents to the system memory. Between cache flushes, the main system memory actually becomes incoherent, because each CPU still has its own private copy of small portions of it. As time passes, these copies may slightly deviate from one another due to cache memory writes. Until the next cache flush or the next system memory write, the most recent data values reside in the cache memories of the individual CPUs.
This creates a memory coherence problem. How can a system memory access (access to outdated data) be prevented from some other system device before it is correctly updated? One solution is called bus snooping. Bus snooping permits a processor to monitor the memory addresses placed on the system bus by other devices. The snooping processor is looking for memory addresses on the system bus that it has cached. When it finds a match, it writes the values of those memory addresses from its cache to the system memory prior to completion of the current bus cycle. Bus snooping is considered to be a critical mechanism in maintaining data coherency. While one processor is accessing memory, another processor snoops on bus activity. If current memory access operations are related to its memory space, the appropriate measures are taken to ensure that all affected processors and bus-masters have the most recent data.

Is maintaining a coherent memory more easily accomplished by removing the cache from multiprocessors and eliminating the need for bus snooping? A close examination of this idea reveals that the implementation of bus snooping and the use of CPU cache are more efficient, allowing SMP systems to realize the full potential of all their processors. In a single-CPU machine, a superfast CPU cache is used primarily as an inexpensive performance booster. However, in a multiprocessing architecture, the main priority is preventing the system bus from becoming a bottleneck, not improving a single CPU's memory performance. When the system relies primarily on the system bus, each additional CPU in the multiprocessing system increases its strain. Therefore, cache memory and bus snooping are considered more important for multiprocessing than for single-processing systems. Well-engineered multiprocessor technology implements a snooping mechanism that is sensitive to the system's performance. The snooping channel must be capable of providing a processor with snoop information even while that processor transfers data. A processor should also be able to broadcast snoop information even while simultaneously receiving data, providing data concurrency and enhanced system performance. This is superior to non-split shared-bus formats, where snooping is limited to the current access on the system bus. In such situations, concurrent data transfers must somehow be related to current snoop activity. If a processor performs a memory transaction, the following questions must be answered:

➤ Is the requested data contained in the requestor's cache, and if so, is it stale or accurate data?

➤ If the data is stale, is the accurate data in main memory, or in another processor's cache?

➤ If the data is in another processor's cache, has the other processor recently changed the data?
The analysis and actions that move and update the data to maintain coherency are part of the bus-snooping process. Bus snooping and the maintenance of data coherency are shared responsibilities of the system processors, core logic, and participating bus-masters. For example, observe the bus-snooping arrangement for an AMD-760 processor in Figure 3.2.

![Figure 3.2](image)

**Figure 3.2** A bus-snooping arrangement—Processor 0 to Processor 1.

A memory request (MR) is launched by processor 0 to the system controller to obtain data or instructions that are not currently in its cache. The system controller interprets this as a snoop request (SR) and then queries processor 1 to determine if it has the requested data. Using its processor-to-system bus, processor 1 will respond to the request. Processor 1 will return the data to processor 0 if it has it. Otherwise, the system controller will have to fetch the data from main memory. While this virtual snooping channel has been created between processor 0, processor 1, and the system controller, processor 0 can concurrently receive messaging on its system-to-processor bus and data on its data bus. Processor 1 can concurrently transmit messaging and data over its processor-to-system and data buses. Notice that transfers unrelated to the current snoop activity can be concurrently performed by both processors. This concurrency plays a significant role in improved performance over less robust multiprocessing architectures. The reverse snooping procedure is shown in Figure 3.3.
The system controller also needs to monitor any transactions occurring across the PCI bus as part of the snooping process. Therefore, the PCI bus-masters also have the power to access processor cache memory areas. The system controller then is responsible for snooping PCI traffic and generating snoop requests to the processors. The virtual snooping channel created between the PCI bus and processor 0 is illustrated in Figure 3.4.

Although maintaining coherent memory remains an important consideration, caching reads and writes prevents the system bus from becoming overloaded. The standard coherent multiprocessing architecture for systems that share the same system buses among multi-CUs is the shared memory architecture. Although the shared-bus design provides better performance at less expense than other multiprocessing architectures, it only scales well up to 32 CPUs, depending on the system and the particular CPU being utilized.

Not all OSs can take full advantage of the concurrency offered by SMP hardware. A suitable NOS must be capable of functioning with components naturally suited to SMP environments. The high degree of parallelism used in these systems permits many different components to run concurrently, effectively utilizing the benefits derived from having many processors available.
SMP processor cores lie in close proximity to each another, being physically connected by a high-speed bus. Although resources such as memory and peripherals are shared by all CPUs, the NOS coordinates the execution of simultaneous threads among them, scheduling each CPU with independent tasks for true concurrency. This permits the simultaneous execution of multiple applications and system services. The only incremental hardware requirements for a true symmetric multiprocessing environment are the additional CPUs, as shown in Figure 3.5. Because the SMP hardware transparently maintains a coherent view of the data distributed among the processors, software program executions do not inherit any additional overhead related to this.
SMP is more than just multiple processors and memory systems design because the NOS can schedule pending code for execution on the first available CPU. This is accomplished by using a self-correcting feedback loop that ensures all CPUs an equal opportunity to execute an specified amount of code; hence, the term symmetric multiprocessing. Only one queue runs in an SMP operating system, with the dumping of all work into a common funnel. Commands are distributed to multiple CPUs on the basis of availability in a symmetric fashion.

Although SMP can potentially use CPU resources more efficiently than asymmetric multiprocessing architectures, poor programming can nullify this potential. Therefore, process scheduling for SMP requires a fairly complex set of algorithms, along with synchronization devices as spin locks, semaphores, and handles. In using all of the available CPU resources, SMP pays a price for the scheduling overhead and complexity, resulting in less than perfect scaling. A system with two CPUs cannot double the overall performance of a single processor. Typically, the second CPU improves the overall performance by about 95 to 99%. With increased overhead for each additional processor, the efficiency goes down, moreso when multiple nodes are involved, as shown in Figure 3.6.

**Figure 3.6** A multiple-node SMP arrangement.

**SMP Derivatives**

Two major derivatives of SMP exist: clustering and *massively parallel processing (MPP)*.
Although the overall goal of both SMP derivatives is speed, each format solves a number of additional problems. Recall how clustering links individual computers, or nodes, to provide parallel processing within a robust, scalable, and fault-tolerant system. These clustered nodes communicate using high-speed connections to provide access to all CPUs, disks, and shared memory to a single application. The clustering applications are distributed among multiple nodes, so that when one machine in the cluster fails, its applications are moved to one of the remaining nodes.

The downside of pure clustering involves the overhead required to coordinate CPU and disk drive access from one machine to the next. This overhead prevents clusters from scaling as efficiently as pure SMP, which maintains closer ties between the processors, memory, and disks. System management is greatly simplified with clustering, however, due to the flexibility of merely adding another node to the cluster for additional processing power. If one machine fails, it’s replaced without affecting the other nodes in the cluster.

Massively parallel processing, where tightly coupled processors reside within one machine, provides scalability far beyond that of basic SMP. Basic SMP suffers from decreasing performance as more processors are added to the mix, whereas MPP performance increases almost uniformly as more processors are introduced. In the MPP arrangement, the processors do not share a single logical memory image and, therefore, do not have the overhead required to maintain memory coherency or move processes between processors. Special applications are needed to tap the power of MPP systems, though, capable of making heavy use of threads, and able to partition the workload among many independent processors. The application must then reassemble the results from each processor. Consider Figure 3.7.

Figure 3.7  An MPP solution.

Early MPP systems were built as single computers, with many processors. A recent trend in computer research, however, is to tap the power of individual machines to build large distributed MPP networks. Examples include the Search for Extraterrestrial Intelligence (SETI), and several Internet cryptography projects. These projects centrally manage and distribute the processing load across multiple machines around the world, and collect the results to build a final answer.
Multiprocessing Advantages

Because multiprocessing targets database and transactional systems, its typical deployment occurs on server boards carrying 4 to 8 processors. Often, high-end systems will feature up to 32 processors working together to scale the workload. Using multiple threads with multiple transactions, multiprocessing systems manage large data-intensive workloads without executing demanding mathematical calculations. An interesting fact regarding the MP’s capability to manage large amounts of data emerges as the amounts of data increase. For example, Table 3.1 indicates the results of testing a multiprocessor configuration against a dual-processor arrangement using a 3GB database.

<table>
<thead>
<tr>
<th>System Configuration</th>
<th>Transactions/second</th>
</tr>
</thead>
<tbody>
<tr>
<td>2 x Intel Xeon processor (2.8MHz)</td>
<td>1260</td>
</tr>
<tr>
<td>2 x Intel Xeon processor MP (2.0MHz)</td>
<td>1102</td>
</tr>
<tr>
<td>4 x Intel Xeon processor MP (2.0MHz)</td>
<td>2070</td>
</tr>
</tbody>
</table>

Observe that although the dual processors are running 40% faster that the multiprocessors, their results are only 14% faster than when pitted against the two MP processors. This is attributed to the level 3 caches being utilized in the two MPs. When the scalability factors come into play using four MP processors, throughput for the multiprocessor design increases by 88% over the two-chip operation. Because the multiprocessor is designed for use with much larger databases than used in Table 3.1, this test can be repeated on a larger system, with even more convincing results. In Table 3.2, the test is repeated using much greater volumes of data.

<table>
<thead>
<tr>
<th>System Configuration</th>
<th>Transactions/second</th>
</tr>
</thead>
<tbody>
<tr>
<td>2 x Intel Xeon processor (2.8MHz)</td>
<td>1433</td>
</tr>
<tr>
<td>2 x Intel Xeon processor MP (2.0MHz)</td>
<td>1485</td>
</tr>
<tr>
<td>4 x Intel Xeon processor MP (2.0MHz)</td>
<td>2497</td>
</tr>
</tbody>
</table>

With a larger database to handle, even the two slower MP processors handle a greater number of transactions than the two DP processors running at 2.8GHz. When using four MP processors, the scalability is maintained.
SMP Commercial Advantages
The commercial advantages to the use of symmetrical multiprocessing include

➤ Increased scalability to support new network services, without the need for major system upgrades
➤ Support for improved system density
➤ Increased processing power minus the incremental costs of support chips, chassis slots, or upgraded peripherals
➤ True concurrency with simultaneous execution of multiple applications and system services

For network operating systems running many different processes operating in parallel, SMP technology is ideal. This includes multithreaded server systems, used for either storage area networking, or online transaction processing.

Multiprocessing Disadvantages
Multiprocessing systems deal with four problem types associated with control processes, or with the transmission of message packets to synchronize events between processors. These types are

➤ Overhead—The time wasted in achieving the required communications and control status prior to actually beginning the client’s processing request
➤ Latency—The time delay between initiating a control command, or sending the command message, and when the processors receive it and begin initiating the appropriate actions
➤ Determinism—The degree to which the processing events are precisely executed
➤ Skew—A measurement of how far apart events occur in different processors, when they should occur simultaneously

The various ways in which these problems arise in multiprocessing systems become more understandable when considering how a simple message is interpreted within such architecture. If a message-passing protocol sends packets from one of the processors to the data chain linking the others, each individual processor must interpret the message header, and then pass it along if the packet is not intended for it. Plainly, latency and skew will increase with each pause for interpretation and retransmission of the packet.
When additional processors are added to the system chain (scaling out), determinism is adversely impacted.

A custom hardware implementation is required on circuit boards designed for multiprocessing systems whereby a dedicated bus, apart from a general-purpose data bus, is provided for command-and-control functions. In this way, determinism can be maintained regardless of the scaling size.

In multiprocessing systems using a simple locking kernel design, it is possible for two CPUs to simultaneously enter a test and set loop on the same flag. These two processors can continue to spin forever, with the read of each causing the store of the other to fail. To prevent this, a different latency must be purposely introduced for each processor. To provide good scaling, SMP operating systems are provided with separate locks for different kernel subsystems. One solution is called fine-grained kernel locking. It is designed to allow individual kernel subsystems to run as separate threads on different processors simultaneously, permitting a greater degree of parallelism among various application tasks.

A fine-grained kernel-locking architecture must permit tasks running on different processors to execute kernel-mode operations simultaneously, producing a threaded kernel. If different kernel subsystems can be assigned separate spin locks, tasks trying to access these individual subsystems can run concurrently. The quality of the locking mechanism, called its granularity, determines the maximum number of kernel threads that can be run concurrently. If the NOS is designed with independent kernel services for the core scheduler and the file system, two different spin locks can be utilized to protect these two subsystems. Accordingly, a task involving a large and time-consuming file read/write operation does not necessarily have to block another task attempting to access the scheduler. Having separate spin locks assigned for these two subsystems would result in a threaded kernel. Both tasks can execute in kernel mode at the same time, as shown in Figure 3.8.

Notice that while Task 1 is busy conducting its disk I/O, Task 2 is permitted to activate the high-priority Task 3. This permits Task 2 and Task 3 to conduct useful operations simultaneously, rather than spinning idly and wasting time. Observe that Task 3's spin lock is released prior to its task disabling itself. If this were not done, any other task trying to acquire the same lock would idly spin forever. Using a fine-grained kernel-locking mechanism enables a greater degree of parallelism in the execution of such user tasks, boosting processor utilization and overall application throughput. Why use multiple processors together instead of simply using a single, more powerful processor to increase the capability of a single node? The reason is that various roadblocks currently exist limiting the speeds to which a single processor can be exposed. Faster CPU clock speeds require wider buses for
board-level signal paths. Supporting chips on the motherboard must be developed to accommodate the throughput improvement. Increasing clock speeds will not always be enough to handle growing network traffic.

![Figure 3.8 A spin lock using a threaded kernel.](image)

### Installing Multiple Processors

Normally, processor manufacturers issue minor revisions to their devices over their manufacturing life cycle. These revisions are called *steppings*, and are identified by special numbers printed on the body of the device. More recent steppings have higher version numbers and fewer bugs. Hardware steppings are not provided in the same manner as software patches that can be used to update a version 1.0 of a program to a 1.01 or 1.02 version. When new steppings come out having fewer bugs than their predecessors, processor manufacturers do not normally volunteer to exchange the older processor for the newer one. Clients are normally stuck with the original stepping unless the updated processor version is purchased separately.
When using multiple processors, always ensure that the stepping numbers on each one match. A significant gap in stepping numbers between different processors running on the same server board greatly increases the chances of a system failure. The server network administrator should make certain that all processors on a server board are within at least three stepping numbers of other processors in the system. Within this safety range, the BIOS should be able to handle small variances that occur between the processors. However, if these variances are more significant, the BIOS may not be capable of handling the resulting problems.

Two methods can be used to determine the stepping of a given Intel processor:

➤ The CPUID utility program
➤ The S-spec number

The *Frequency ID utility* identifies a genuine Intel processor, and determines the processor’s family, model, and stepping. For a given processor, the Frequency ID utility and its installation instructions can be downloaded from Intel’s website at: [http://support.intel.com/support/processors/tools/frequencyid/](http://support.intel.com/support/processors/tools/frequencyid/).

The *S-spec number* is located on the top edge of the specified Intel processor, as shown in Figure 3.9. It identifies the specific characteristics of that processor, including the stepping. The S-spec number can be referenced in the specification update for the selected processor, and downloaded from Intel’s website.

Another common but avoidable problem occurs with multiprocessor server boards when processors are installed that are designed to use different clocking speeds. To expect multiprocessors to work properly together, they should have identical speed ratings. Although it’s possible that in some cases a faster processor can be clocked down in order to work in the system, problems normally occurring between mismatched processors will cause operational...
glitches beyond the tolerance of server networks. The safest approach with multiprocessor server boards is to install processors that are guaranteed by their manufacturer to work together. These will generally be matched according to their family, model, and stepping versions.

Figure 3.9 The S-spec number location for an Intel processor.

**Server Processor Subsystems**

Server+ Exam Objective 1.14 states that the test taker should understand the processor subsystem of a server. This understanding includes being able to describe

➤ Multiprocessing systems
  ➤ What are they?
  ➤ How do they differ from dual-processor systems?

➤ 64-bit server environments
  ➤ What are they?
  ➤ Why and when are they important?
  ➤ What are the different architectures?

In determining when and why 64-bit server environments are important, the test taker should consider:

➤ Microprocessor bit ratings
➤ Database management software
➤ WOW64
➤ Windows Server 2003
➤ Redhat Linux
➤ Windows XP Professional x64 Edition
➤ Internal registers
➤ 64-bit architecture

Know that Redhat Linux and Windows Server 2003 operating systems are available in 64-bit versions.

Server Environments for 64-Bit Processors

When examining information about PCs and servers, their applicable microprocessors are usually rated according to a number of bits, such as 32-bit or 64-bit. This rating is a tipoff as to how powerful or capable its associated computer system is, because a bit represents the smallest unit of information that can exist in a digital environment. A 32-bit rating means that 32 bits of digital data can be processed or transmitted in parallel, for each clock cycle. This rating can also refer to the number of bits used for a single element in the server’s internal data structure.

When used in conjunction with a processor, this term indicates the width of its internal registers, which compose a special, high-speed storage area. All data must be represented in a register before being processed in any way. In order to multiply two numbers, both of the numbers must first reside in specified registers. After multiplication, the result must also be placed in a register. These registers can alternately contain specified memory address locations, where the actual data is stored. The number and size of a given microprocessor’s registers determine its power and its speed. For a 32-bit microprocessor, each internal register is 32 bits wide and each program instruction manipulates 32 bits of data. Obviously, such a processor would also process data and memory addresses represented by 32 bits. The Intel 486 microprocessor, shown in Figure 3.10, is a well-known example of a 32-bit CPU.

Figure 3.10  The Intel 486 microprocessor.
A 64-bit rating refers to a processor containing registers that store 64-bit numbers, indicating that a 64-bit server board architecture would process double the amount of data processed for each clock cycle, compared to a 32-bit system. Clients experience a noticeable performance increase due to the ability of the 64-bit microprocessor to handle more memory and larger files. One of the most important benefits of using 64-bit processors is the amount of memory supported by the accompanying system architecture, which is capable of addressing an impressive 1 terabyte (1,000GB) of memory. Compare this with a contemporary 32-bit desktop system, where the maximum amount of RAM supported is 4GB. Many commercial motherboards cannot support even that much RAM, which is shared between any running applications and the operating system (OS) software.

It should be noted that the majority of desktop computers today do not contain as much as 4GB of installed memory. Such amounts are not required to properly use most small business and home desktop computer software. In residential systems, 32-bit operations provide more than adequate processing power, although it is possible in the near future that 4GB of RAM could become a limitation, as more complex software and animated graphical games evolve. However, in scientific and data management industries, the 4GB memory limitations of the 32-bit system have already been reached. Because the need for 64-bit processing has become apparent, major database management software developers, such as Oracle and Microsoft, currently offer 64-bit versions of their database management systems.

The momentum for developing 64-bit systems has come from the commercial networking environment, where larger, faster, more capable servers are in constant demand. The development of 64-bit processors, such as the Intel Itanium has been paralleled by the development of other key components, such as 64-bit operating systems, software, and drivers capable of taking advantage of 64-bit processor features.

These developments provide real benefits to demanding applications such as video encoding, scientific research, and massive database management, where being able to load massive amounts of data into the network server’s RAM memory is required. AMD has also developed a 64-bit microprocessor capable of handling these heavy-duty applications.

In supercomputing and in database management systems, 64-bit architecture has been used for many years. Having to access huge amounts of data, many companies and organizations have already made the transition to 64-bit servers. They support a greater number of larger files, and are capable of effectively loading large enterprise databases into RAM memory for faster searches and data retrieval. By using 64-bit servers, these organizations are
Chapter 3

The capabilities of the x86 architecture have been extended, enabling customers using a 64-bit operating system to seamlessly run 32-bit and cutting edge 64-bit applications. The newer hardware can also be used with today’s standard 32-bit operating systems and applications. Microsoft has developed a new 64-bit version of Windows XP Professional called Windows XP Professional x64 Edition.

Windows XP Professional x64 Edition permits users to take advantage of the new 64-bit technology to improve PC or workstation performances. It provides a rich platform on which to integrate both new 64-bit applications and current 32-bit applications using what is called the Windows on Windows 64 (WOW64) x86 emulation layer. The WOW64 subsystem isolates 32-bit applications from 64-bit applications, in order to prevent file and registry collisions. The unique feature about Windows XP Professional x64 Edition processor architecture is its ability to run both 32-bit and 64-bit applications equally well.

Server Software Architectures for 64-Bit Processors

Windows XP Professional x64 Edition is designed to address the most demanding business needs of technical workstation users who require large amounts of memory and floating-point performance in areas such as mechanical design and analysis, digital content creation, and scientific and high-performance computing applications. The following business segments should benefit from Windows XP Professional x64 Edition.

➤ Engineering (CAD/CAM)—Automotive or aerospace design engineers conceptualize designs while meeting stringent design safety requirements. Designers and engineers who use computer-aided design (CAD) and engineering applications will benefit from the large memory support, fast memory throughput, and improved floating-point speeds provided by 64-bit systems. They need these applications to work with
larger models in a shorter period of time.

➤ Digital Content Creation—Two-dimensional (2D) and three-dimensional (3D) animation and rendering, video editing, and game development are its three major areas. It allows game developers and animators to save time in rendering models or scenes. The ability to view completely rendered models during the development process gives animators and developers the freedom to work at their peak level of creativity.

➤ 3D Gaming—Video game developers are at work building exciting 64-bit native games to help them push these limits even further because they hold the potential for significantly accelerated graphics rendering. Game makers build today's most sophisticated 3D titles around extremely complicated database engines that enable the use of artificial intelligence, and the play of massive-level 3D games, in ways that were not formerly possible.

➤ Video Editing—Professional and amateur photographers, web designers, and home PC enthusiasts are increasingly using PCs to do sophisticated video editing and photo manipulation, especially with 64-bit workstations.
Exam Prep Questions

1. Which of the following items is not considered important when matching characteristics for additional Intel microprocessors on SMP-capable server boards?
   - A. Family
   - B. Stepping
   - C. Model
   - D. Cache

2. The NOS is capable of running multiple CPUs. However, following the installation of the newest processor, the system boots up without recognizing it. What is the most sensible action to take at this point?
   - A. Configure the NOS for SMP.
   - B. Replace the bad CPU.
   - C. Examine the stepping of the new CPU.
   - D. Perform a BIOS upgrade.

3. Select the most accurate statement:
   - A. SMP systems are more expensive to implement than ASMP.
   - B. SMP is easier to implement than ASMP.
   - C. SMP is a slower implementation than ASMP.
   - D. In an SMP server system, all CPUs share the same board, but require their own memory, peripherals, and NOS.

4. What is the most important functional aspect of SMP?
   - A. The running of multiple threads simultaneously on multiple processors
   - B. The division of tasks between multiple processors being executed one at a time
   - C. The running of multiple NOSes by multiple processors
   - D. The composition of multiple threads into a single thread prior to execution

5. Which of the following is not a 16-bit or a 32-bit operating system?
   - A. Panther
   - B. Windows 2000 Server
   - C. Windows Server 2003
   - D. Windows 98
Exam Prep Answers

1. Answer D is the correct answer. The cache memory is not considered when matching characteristics between processors on SMP server boards. The important characteristics for matching include the family, model, and stepping. In addition, if these characteristics match, it’s likely the cache will also. Answers A, B, and C are therefore all incorrect.

2. Answer A is the correct answer. Although Answer D might be necessary in some situations, the most likely cause of the problem is that the NOS is not properly configured to provide SMP support. Answer B is incorrect because not enough information is known to assume that the new CPU is bad. Answer C would usually be considered before selecting the new CPU to ensure compatible stepping values.

3. Answer B is the correct answer. ASMP requires specialized knowledge to implement its computer, whereas symmetrical multi-processing is relatively easy to implement. Answer A is incorrect because an SMP solution is less expensive to implement. Answer C is incorrect because SMP is faster than a single-processor solution. Answer D is incorrect because SMP processors share not only the same board, but the NOS, memory, and peripherals as well.

4. Answer A is the correct answer. SMP time-slices its operations into multiple threads for simultaneous execution by multiple processors. Answer B is incorrect because nothing is gained by having multiple processors that can only execute one task at a time. Answer C is incorrect because running multiple NOSes by multiple processors does not describe SMP. Answer D is incorrect because SMP does just the opposite.

5. Answer C is the correct answer. The Windows Server 2003 OS package provides 64-bit computing for business and home use. Answer A is incorrect because Apple’s Panther OS uses a 32-bit platform. Answer B is incorrect because Windows 2000 Server is also a 32-bit system. Answer D is incorrect because Windows 98 is a 16-bit platform capable of running some 32-bit software.