The BIOS of a multiple processor system passes partial control to the operating system following an abbreviated power-on initialization. The BIOS program retains control of one or more processors and any memory that was untested during power-on. While the operating system is loaded and the system is operational, the BIOS program tests the retained memory. The BIOS program then relinquishes control of the retained memory and processor(s) during runtime to the operating system.
BEGIN

PERFORM MINIMAL INITIALIZATION

IDENTIFY MINIMUM MEMORY

INITIALIZE & TEST MINIMUM MEMORY

INITIALIZE REMAINDER OF SYSTEM

PASS PARTIAL CONTROL TO OPERATING SYSTEM

A

FIGURE 3
BEGIN

DETERMINE WHICH PROCESSOR(S) TO RETAIN

DETERMINE AMOUNT OF MEMORY TO RETAIN

IDENTIFY AVAILABLE PROCESSORS & MEMORY IN TABLE

FIND OPERATING SYSTEM IMAGE

LOAD IMAGE INTO RELINQUISHED MEMORY

JUMP TO OPERATING SYSTEM ENTRY POINT

DONE

FIGURE 4
POWER-ON ROUTINE:

1. DETECT REMAINING MEMORY
2. DIVIDE REMAINING MEMORY INTO PORTIONS
3. INITIALIZE & TEST A PORTION OF MEMORY
4. RELINQUISH PORTION TO THE OPERATING SYSTEM

OPERATING SYSTEM:

1. PERFORM INITIALIZATION
2. PRIMARY SYSTEM SERVICES ARE AVAILABLE

If MORE INITIALIZED UNPREPARED MEMORY, go back to 242. Otherwise, RELINQUISH PROCESSOR TO THE OPERATING SYSTEM.

DONE

FIGURE 5
BEGIN

IDENTIFY SYSTEM COMPONENTS BY READING CONFIGURATION TABLE

INITIALIZE SYSTEM COMPONENTS

RECEIVE INTERRUPT FROM BIOS PROGRAM

IDENTIFY NEW SYSTEM COMPONENTS

DONE

FIGURE 6
MULTI-STAGED BIOS-BASED MEMORY TESTING

BACKGROUND

[0001] This invention relates to basic input/output systems (BIOS) programs and, more particularly, to improved memory testing.

[0002] In a processor-based system, powering on the system is commonly known as a “boot” process. The boot process is performed by a program, usually located in a read-only memory (ROM) or flash memory of the processor-based system. The ROM/flash program includes a software program known as the basic input/output system, or BIOS. The BIOS program executes commands such that the various circuitry of the processor-based system may be initialized. The BIOS program also includes functions that may be used by software, including operating systems, for communicating with different devices in the processor-based system.

[0003] Upon receiving power to the processor-based system, the BIOS program immediately begins execution. The BIOS performs initialization functions, such as detecting and testing the memory, the display, the non-volatile media, such as hard disk and floppy disk drives, and so on. In some systems, an abbreviated power-on, or “quick-boot,” may be available, which may test some, but not all circuitry.

[0004] Once the BIOS program completes initialization and testing of the system, control is typically transferred to an operating system, usually located on the hard disk drive. Once the operating system gains control of the processor-based system, all run-time operations, including any execution of application programs, are controlled by the operating system. The operating system may or may not utilize the BIOS functions in communicating with the hardware of the processor-based system.

[0005] Typically, the BIOS program performs minimal system initialization, followed by memory sizing, error-correcting code (ECC) memory filling if required, and varying levels of memory testing. Once the memory is deemed sound by the program, the remainder of the system is initialized. Only after all these memory-related tasks are complete will the BIOS program transfer control of the system, including processors, memory, and input/output (I/O), to the operating system. This serial approach to initialization results in delays before the system is available for runtime operation.

[0006] The current memory technology allows for very inexpensive processor-based systems to nevertheless include a large amount of memory. These very large memory arrays may create significant delay during system initialization. Exhaustive tests on large amounts of memory may take 10, 20, or even 30 minutes to complete. Where the memory is tested in the BIOS, the system is required to be “off-line” during the memory testing.

[0007] Thus, there is a continuing need to quickly boot an operating system on a processor-based system while still performing a thorough test of the memory.

BRIEF DESCRIPTION OF THE DRAWINGS

[0008] FIG. 1 is a block diagram of a system according to one embodiment of the invention;

[0009] FIG. 2 is a state diagram illustrating operation of the BIOS according to one embodiment of the invention;

[0010] FIG. 3 is a flow diagram illustrating operation of the system according to one embodiment of the invention;

[0011] FIG. 4 is a flow diagram illustrating how the BIOS passes partial control to the operating system according to one embodiment of the invention;

[0012] FIG. 5 is a flow diagram illustrating parallel operation between the BIOS and operating system programs according to one embodiment of the invention;

[0013] FIG. 6 is a flow diagram illustrating operation of the operating system according to one embodiment of the invention; and

[0014] FIG. 7 is a functional block diagram of a system according to one embodiment of the invention.

DETAILED DESCRIPTION

[0015] In FIG. 1, according to one embodiment, a system 100 includes two or more processors 10. The processors 10 may include the PENTIUM® family of processors and the 80x86 families of processors from Intel Corporation, as examples.

[0016] The system 100 further includes a memory 20 that may be divided into portions. The memory 20 may include random access memory (RAM), dynamic random access memory (DRAM), synchronous DRAM (SDRAM), static ram (SRAM), or Rambus® DRAM (RDRAM), as examples.

[0017] In one embodiment, software for both power-on and run-time operation of the system 100 are included. A BIOS program 14 executes as the system 100 receives power. As explained above, the BIOS program 14 performs initialization and testing of components within or connected to the system 100. Such components may include video and audio devices, mass storage media, keyboard and mouse circuitry, serial and parallel ports, memory, and so on.

[0018] In one embodiment, the BIOS program 14 resides in a storage 12 such as a read-only memory (ROM). In a second embodiment, the storage 12 is a flash memory device. In other embodiments, a different power-on program that is not BIOS-based may be utilized in performing the operations described herein.

[0019] For run-time operation, the system 100 further includes an operating system program 16. In one embodiment, the operating system program 16 resides in a non-volatile storage device 18, such as a hard disk drive or compact disk (CD) ROM. In a second embodiment, the operating system program 16 does not reside on the system 100. Instead, the system 100 is accessible to a network (not shown) by a network interface card 26. Once the network connection is made, the operating system software 16 may be downloaded to the system 100.

[0020] In one embodiment, the BIOS program 14 retains control of a portion of the system 100 instead of relinquishing full control of the system 100 to the operating system 16. This enables the BIOS program 14 to minimally configure and test the system 100 such that the operating system may be “booted” more expeditiously. Further, the portion of the system that was not relinquished to the operating system 16
may be fully initialized and tested during runtime, e.g., after the system 100 is fully capable.

[0021] Accordingly, in FIG. 1, the processor 10a is deemed a “retained” processor, e.g., that processor that is retained by the BIOS program 14. The processor 10b is known as a “relinquished” processor, as one relinquished to the operating system 16 by the BIOS program 14. More than one processor 10a may be retained and more than one processor 10b may be relinquished. Like the processors 10, the memory 20 may be divided into retained memory portion 20a and relinquished memory portion 20b.

[0022] In one embodiment, the system 100 further includes a configuration table 22, which is accessible to both the BIOS program 14 and the operating system program 16. The configuration table 22 includes information about resources within and connected to the system 100. In particular, the configuration table 22 supplies the operating system 16 with the amount of memory 20 available in the system 100 as well as the number and type of processors 10.

[0023] Typically, the BIOS of a processor-based system performs initialization of system components, including memory. Memory initialization may include sizing the memory, ECC filling, and various forms of memory testing. Only after these tasks are complete will the BIOS transfer control of the system, including processors, memory, and input/output (I/O) to the operating system. Particularly where the system has large amounts of memory, this serial approach to initialization delays runtime use of the system.

[0024] Recently, hardware, software, and operating system technologies have evolved to allow systems to “hot add,” “hot remove,” and “hot replace” system components. Using such technology, processors and memory, for example, may be added and removed while a system is running.

[0025] For example, under the Advanced Configuration and Power Interface (ACPI) Specification, processors may be ejected or dynamically inserted into a system during runtime operation. Likewise, other hardware, such as memory, may be “hot added” or “hot removed.” The ACPI Specification, revision 2.0, was published by a consortium (Compaq Computer Corporation, Intel Corporation, Microsoft Corporation, Phoenix Technologies, Ltd., and Toshiba Corporation), on Jul. 27, 2000.

[0026] In one embodiment, the system 100 has the ability to dynamically add resources to and remove resources from the system 100 during runtime operation. In one embodiment, the BIOS program 14 and the operating system 16 implement the dynamic addition or removal of resources according to the ACPI Specification, Revision 2.0. Under ACPI, the addition of a device to the system 100 causes an interrupt to be sent to the operating system 16.

[0027] The operating system 16 is then informed, through an interpreted language, known as ACPI Source Language, or ASL, of the additional device. Briefly, the ASL language includes device and method constructs, wherein the device construct identifies the device being added (e.g., processor, memory) and the method construct describes attributes of the device. Thus, in one embodiment, the operating system 16 determines the basis for the interrupt using the ASL code. From this information, the operating system 16 may locate the device. In addition to memory or processor, the device may be a storage, a network interface card, and so on.

[0028] Using this dynamic add and remove capability, according to one embodiment, the BIOS program 14 relinquishes control of less than all resources of the system 100 to the operating system program 16. Instead, the BIOS program 14 retains control of one or more processors, such as the processor 10a, so that the BIOS program 14 may continue executing after the operating system program 16 has been loaded and executed.

[0029] By retaining one or more processors 10 the BIOS program 14 may execute simultaneously with the operating system program 16, each using distinct processors 10. In other words, the BIOS program 14 may use the retained processor(s) 10a while the operating system 16 uses the relinquished processor(s) 10b.

[0030] The retained processor(s) 10a and the relinquished processor(s) 10b is share resources in the system 100, including the memory 20. Accordingly, arbitration between the two or more processors 10 for control of system resources is handled just as in any system employing multiple processors that execute simultaneously. In one embodiment, a North Bridge controller is used to arbitrate between the use of system resources (see FIG. 7).

[0031] Further, in one embodiment, the BIOS program 14 retains a portion of the memory 20 (retained memory 20a) while relinquishing control of the remainder of the memory 20 (relinquished memory 20b) to the operating system 16. In one embodiment, the relinquished memory 20b includes enough of the total memory 20 to enable the system 100 to be fully functional. As will be shown, below, the BIOS program 14 initializes and tests the retained memory 20a.

[0032] Following the memory test, the retained memory 20a may be released to the operating system 16, or, in other words, become relinquished memory 20b.

[0033] In one embodiment, the BIOS program 14 releases the memory 20 in stages, e.g., tests a first portion of retained memory 20a, releases the first portion, tests a second portion of retained memory 20a, releases the second portion, tests a third portion of retained memory 20a, releases the third portion, and so on. In a second embodiment, the BIOS program 14 tests all the retained memory 20a first, then releases the retained memory 20a to the operating system 16.

[0034] Likewise, in one embodiment, the BIOS program 14 releases the processors 10 once their use is exhausted. Where the retained processors 10a include multiple processors, each testing multiple portions of memory 20a in parallel, a retained processor 10a may be released to the operating system 16 once memory testing for the associated memory 20a is complete. Alternatively, the BIOS program 14 may release all the retained processors 10a to the operating system 16 simultaneously, following completion of the testing of all the retained memory 20a.

[0035] In FIG. 2, a state diagram depicts the operation of the BIOS program 14, according to one embodiment. Assume the BIOS program 14 is executing on a retained processor 10a in order to test retained memory portion 20a. The BIOS program 14 initializes and tests a portion of the retained memory 20a (state 20). Once the test is complete,
the BIOS program 14 relinquishes control of the retained memory portion 20a to the operating system 16 (state 52).

[0036] Where additional retained memory 20a is available for testing, the BIOS program 14 may perform another initialization and test (state 50). Once the initialization and testing of the retained memory 20a is complete, the BIOS program 14 may relinquish control of the retained processor 10a to the operating system 16 (state 54).

[0037] Where the processor 10a, is the last processor retained by the BIOS program 14, total control of the system 10 has been relinquished to the operating system 16. However, where the BIOS program 14 retains additional processors 10a, additional initialization and testing of the remaining retained memory 20a may still be pending, as indicated by the dotted line in the state diagram of FIG. 2.

[0038] Although the memory 20 may take a long time to initialize and test, the BIOS program 14 may retain other resources in order to expedite runtime use of the system 100. Depending on the resources retained, the runtime operation of the system 100 may be limited to fewer than all the operations for which the system 100 is capable. For some operating environments, however, the tradeoff in runtime capability is well worth the expedited availability of runtime operation.

[0039] In FIG. 3, a flow diagram illustrates operation of the BIOS program 14, according to one embodiment, in preparing to pass partial control of the system 100 to the operating system 16. The BIOS program 14 performs minimal system initialization (block 202). The amount and type of system initialization performed may be system-dependent. In one embodiment, the minimal system initialization includes initialization and testing of all processors 10, both retained 10a and relinquished 10b, in the system 100. In a second embodiment, all system resources are initialized and tested except the retained memory 20a.

[0040] The BIOS program 14 further identifies a minimum amount of memory 20 that is to be passed to the operating system 16 (block 204). In one embodiment, the minimum amount of memory 20 is the amount of relinquished memory 20b. The relinquished memory 20b is then initialized and tested (block 206) so that it may be passed to the operating system 16. The BIOS program 14 also initializes the remainder of the system 100 (block 208). Once the partial system is initialized and tested, control is passed to the operating system 16 (block 210).

[0041] In FIG. 4, a flow diagram illustrates how the BIOS program 14 passes partial control of the system 100 to the operating system 16, according to one embodiment. The BIOS program 14 determines which and how many processors 10 to retain (block 222), e.g., not pass on to the operating system 16. The BIOS program 14 also decides how much of the memory 20 to retain (block 224). Such determinations may be based upon the system capability, the environment in which the system operates, and other factors.

[0042] The BIOS program 14 updates the configuration table 22, which is accessible to the operating system 16 (block 226). In one embodiment, the configuration table 22 includes fields for specifying the number of processors 10, along with identifying features for each processor. The configuration table 22 also indicates the amount of memory 20 in the system, as well as the type of memory and other configuration information.

[0043] In one embodiment, the number of processors 10 and the amount of memory 20 listed in the configuration table 22 is less than the number of processors in the system 100. Instead, the configuration table 22 reflects the number of relinquished processors 10b and the amount of relinquished memory 20b (e.g., the number of processors 10b and the amount of memory 20b to be relinquished to the operating system 16). In essence, the BIOS program 14 is lying to the operating system 16. By identifying only the system resources that are to be relinquished to the operating system 16, the BIOS program 14 may retain some resources for its own use.

[0044] Next, according to one embodiment, the BIOS program 14 finds the image of the operating system 16 (block 228). The image may reside on the system 100, such as in the non-volatile storage 18, a compact disk (CD) read-only memory (ROM), or other magnetic medium. Or, the image may be retrievable to the system 100, such as by downloading the image over a network. Once the operating system 16 image is found, in one embodiment, the image is loaded into a portion of the relinquished memory 20b (block 230).

[0045] According to one embodiment, the BIOS program 14 jumps to an entry point in the image (block 232), which commences the execution of the operating system 16. The BIOS program 14, however, continues to run on one of the retained processors 10a of the system 100.

[0046] In FIG. 5, the parallel operation of the BIOS program 14 and the operating system 16 is illustrated in a flow diagram. In one embodiment, the parallel operations begin when the BIOS program 14 jumps to an entry point in the image (block 232 of FIG. 4). Because the BIOS program 14 retains one or more processors 10a, the BIOS program 14 may continue executing.

[0047] In the sequence of FIG. 5, the BIOS program 14 retains only one processor 10. Accordingly, the flow diagram depicts two parallel threads: one for the BIOS program 14 and one for the operating system. However, where the BIOS program 14 retains more than one processor, additional BIOS program 14 threads may operate simultaneously in testing the retained memory 20b of the system 100.

[0048] In one embodiment, the BIOS program 14 detects the retained memory 20b (block 240) and divides the retained memory 20b into portions (block 242). A portion of the memory 20b is initialized and tested (block 244). In one embodiment, once the testing is complete, the portion is relinquished to the operating system 16 (block 246).

[0049] The BIOS program 14 determines whether more retained memory 20b is present in the system 100 (diamond 248). If so, another portion of the retained memory 20b is initialized and tested (block 244). Otherwise, the retained processor 10a is relinquished to the operating system 16 (block 250).

[0050] While the BIOS program 14 is testing the retained memory 20a, the operating system 16 may perform its own system initialization (block 260). The primary system services become available (block 262), allowing the system 100 to be fully utilized for runtime operation.

[0051] In one embodiment, to relinquish control of the processor 10 or the memory 20, the BIOS program 14 sends
an interrupt to the operating system 16. The ACPI specification, revision 2.0, defines methods and events that allow the BIOS program 14 to communicate with the operating system 16. The operating system 16 invokes an interpreted language, such as ASL, that allows the operating system 16 to identify the source of the interrupt: the addition of a new processor to the system. In one embodiment, the operating system 16 receives the retained processor 10c in accordance with the ACPI specification, revision 2.0.

[0052] In FIG. 6, according to one embodiment, the operating system 16 begins execution once the BIOS program 14 has jumped to the operating system entry point (block 232 of FIG. 4). The operating system 16 identifies system components by reading the configuration table 22 (block 282). From the system resource information identified in the configuration table 22, the operating system 16 initializes the system components (block 284). For some devices, the initialization may include testing operations.

[0053] The operating system 16 is fully operational, to the extent that system devices were made available to the operating system 16 by the BIOS program 14. However, the BIOS program 14 may retain some system devices and perform operations simultaneously with the running of the operating system 16, as described, above. Where the BIOS program 14 is ready to relinquish the retained devices, in one embodiment, the BIOS program 14 sends an interrupt to the operating system 16 (block 286).

[0054] In addition to interrupting the operating system 16, the operating system 16 is informed of the newly relinquished device or devices. In one embodiment, the operating system 16 is informed by receiving ACPI constructs (e.g., methods), written in ASL, from the BIOS program 14. These constructs supply information about the new device(s) to the operating system 16. Accordingly, the operating system 16 may determine what new circuitry is being relinquished (block 288). These steps (284, 286, and 288) may repeat until the BIOS program 14 has released all retained resources.

[0055] One benefit of the operations described herein is the reduction of system downtime, in some embodiments. By testing the memory 20 in multiple stages, the operating system 16 gains control of the system 100 sooner that if all the memory 20 was initialized and tested. The downtime savings may be significant, particularly where the system 100 includes a substantial amount of memory 20. Some large systems, for example, may take twenty to thirty minutes to boot.

[0056] A second benefit of these operations is the identification of memory errors, due to the thorough testing of the memory 20 by the BIOS program 14 after the operating system 16 is loaded. Some prior art system avoid the lengthy boot process by running faster memory tests. Unfortunately, these faster tests are often less effective and enable more memory errors to escape unnoticed. By instead utilizing a staged memory test, more exhaustive memory testing may be performed without compromising the availability of the system for runtime use.

[0057] In FIG. 7, according to one embodiment, a functional block diagram of the system 100 includes three processors 10a, 10b and 10c connected to a host bus 62. The host bus 72 may be an Intel PENTIUM® PRO bus, as one example. (“See the Pentium Pro Processor Specification,” update release May 1998.)
retaining control of a second portion of the processor-based system by a power-on program; running both the power-on program and the operating system program simultaneously; retaining a portion of a memory by the power-on program; performing memory operations on the retained portion of the memory by the power-on program while the operating system is running; relinquishing a second portion of the memory to the operating system; and performing memory operations on the relinquished portion of the memory by the operating system while the power-on program is running.

6. A method comprising:
passing control of at least one portion of a processor-based system to an operating system loaded on the processor-based system;
retaining control of a second portion of the processor-based system by a power-on program;
running both the power-on program and the operating system program simultaneously;
retaining a portion of a memory by the power-on program; performing memory operations on the retained portion of the memory by the power-on program while the operating system is running;
performing memory operations on the retained portion of the memory by the power-on program while the operating system is running further comprising:
initializing the retained portion of memory; and
testing the retained portion of memory.

7. The method of claim 3, retaining control of at least one processor of a plurality of processors further comprising:
identifying the plurality of the processors available to the system in a table readable by the operating system;
notifying the operating system that the plurality of the processors are available; and
deciding to include the at least one processor in the table.

8. The method of claim 7, running both the power-on program and the operating system simultaneously further comprising:
executing the power-on program by the at least one processor of the plurality of processor; and
executing the operating system by a second processor, wherein the second processor is identified by the table.

9. A system comprising:
a first storage to store a power-on program;
a second storage to store an operating system;
a first element controlled by the operating system; and
a second element controlled by the power-on program.

10. The system of claim 9, wherein the first element is a first processor and the power-on program executes on the first processor.

11. The system of claim 10; wherein the second element is a second processor and the operating system executes on the second processor.

12. A system comprising:
a first storage to store a power-on program;
a second storage to store an operating system;
a first element controlled by the operating system;
a second element controlled by the power-on program, wherein the first element is a first processor and the power-on program executes on the first processor, wherein the second element is a second processor and the operating system executes on the second processor, and a configuration table accessible to both the power-on routine and the operating system to identify the first element.

13. The system of claim 9, wherein the first element is a first portion of a memory.

14. The system of claim 13, wherein the second element is a second portion of the memory.

15. An article comprising a medium storing instruction to enable a processor-based system to:
pass control of at least one portion of the processor-based system to an operating system loaded on the processor-based system;
retain control of a second portion of the processor-based system by a power-on program; and
run both the power-on program and the operating system program simultaneously.

16. The article of claim 15, further storing instructions to enable a processor-based system to:
retain a portion of a memory by the power-on program; and
perform memory operations on the retained portion of the memory by the power-on program while the operating system is running.

17. An article comprising a medium storing instructions to enable a processor-based system to:
pass control of at least one portion of the processor-based system to an operating system loaded on the processor-based system;
retain control of a second portion of the processor-based system by a power-on program;
run both the power-on program and the operating system program simultaneously;
retain a portion of a memory by the power-on program;
perform memory operations on the retained portion of the memory by the power-on program while the operating system is running;
relinquish a second portion of the memory to the operating system; and
perform memory operations on the relinquished portion of the memory by the operating system while the power-on program is running.
18. The article of claim 17, further storing instructions to enable a processor-based system to:

identify the plurality of processors available to the processor-based system in a table readable by the operating system;

notify the operating system that the plurality of processors are available; and

decide to include at least one processor of the plurality of processors in the table.

19. The article of claim 18, further storing instructions to enable a processor-based system to:

execute the power-on program by the at least one processor of the plurality of processors; and

execute the operating systems by a second processor, wherein the second processor is identified in the table.

20. A method comprising:

receiving control of a portion of a processor-based system by an operating system; and

receiving control of a second portion of the processor-based system while executing an operating system.

21. The method of claim 19, receiving control of a second portion of the processor-based system while executing the operating system further comprising:

receiving an interrupt; and

receiving methods that describe the second portion.

22. An article comprising a medium storing instructions to enable a processor-based system to:

receive control of a portion of the processor-based system by an operating system; and

receive control of a second portion of the processor-based system while executing an operating system.

23. The article of claim 22, further storing instructions to enable a processor-based system to:

receive an interrupt; and

receive methods that describe the second portion

24. A system comprising:

two or more processors;

a device comprising a first portion and a second portion, wherein the device is coupled to the two or more processors; and

a power-on routine executable by one of the two or more processors, wherein the power-on routine relinquishes control of the first portion of the device before loading an operating system.

25. A system comprising:

two or more processors;

a device comprising a first portion and a second portion, wherein the device is coupled to the two or more processors; and

a power-on routine executable by one of the two or more processors, wherein the power-on routine relinquishes control of the first portion of the device before loading an operating system,

wherein the power-on routine further:

tests the second portion of the device while the operating system is running; and

relinquishes control of the second portion of the device to the operating system.

26. The system of claim 25, wherein the device is a memory.

27. The system of claim 25, wherein the device is a group of processors and the first portion of the device is a first processor and the second portion of the device is a second group of processors.

28. A system comprising:

two or more processors;

a device comprising a first portion and a second portion, wherein the device is coupled to the two or more processors; and

an operating system executable by one of the two or more processors, wherein the operating system receives control of the first portion of the device when the operating system is executed.

29. The system of claim 28, further comprising:

a power-on routine to relinquish control of the first portion of the device to the operating system.

30. A system comprising:

two or more processors;

a device comprising a first portion and a second portion, wherein the device is coupled to the two or more processors;

an operating system executable by one of the two or more processors, wherein the operating system receives control of the first portion of the device when the operating system is executed; and

a power-on routine to relinquish control of the first portion of the device to the operating system,

wherein the operating system further:

receives an interrupt from the power-on routine; and

receives control of the second portion of the device.