(12) INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
29 June 2017 (29.06.2017)

WIPO | PCT

(10) International Publication Number
WO 2017/112126 A1

(51) **International Patent Classification**:
G06F 9/44 (2006.01)

(21) **International Application Number**:
PCT/US2016/061846

(22) **International Filing Date**:
14 November 2016 (14.11.2016)

(25) **Filing Language**: English

(26) **Publication Language**: English

(30) **Priority Data**:
14/976,990 21 December 2015 (21.12.2015) US

(71) **Applicant**: **INTEL CORPORATION** [US/US]; 2200 Mission College Boulevard, Santa Clara, California 95054 (US).

(72) **Inventors**: **THIYAGARAJAH, Arumugam**; 1037 Halidon Way, Folsom, California 95630 (US). **KHANNA, Gaurav**; 5551 NE Azores Court, Hillsboro, Oregon 97124 (US). **JEYASINGH, Stalinselvaraj**; 13394 NW Keeton Park Lane, Portland, Oregon 97229 (US). **MEHTA, Sohil**; 805 NE 62nd Ave, Apt Q, Hillsboro, Oregon 97124 (US). **JAGASIA, Mukesh J.**; 250 McAdoo Drive, Apt. 713, Folsom, California 95630 (US).

(74) **Agents**: **AUYEUNG, Al** et al.; 1211 SW 5th Avenue, Suite 1500-1900, Portland, Oregon 97204 (US).

(81) **Designated States** *(unless otherwise indicated, for every kind of national protection available)*: AE, AG, AL, AM, AO, AT, AU, AZ, BA, BB, BG, BH, BN, BR, BW, BY, BZ, CA, CH, CL, CN, CO, CR, CU, CZ, DE, DJ, DK, DM, DO, DZ, EC, EE, EG, ES, FI, GB, GD, GE, GH, GM, GT, HN, HR, HU, ID, IL, IN, IR, IS, JP, KE, KG, KN, KP, KR, KW, KZ, LA, LC, LK, LR, LS, LU, LY, MA, MD, ME, MG, MK, MN, MW, MX, MY, MZ, NA, NG, NI, NO, NZ, OM, PA, PE, PG, PH, PL, PT, QA, RO, RS, RU, RW, SA, SC, SD, SE, SG, SK, SL, SM, ST, SV, SY, TH, TJ, TM, TN, TR, TT, TZ, UA, UG, US, UZ, VC, VN, ZA, ZM, ZW.

(84) **Designated States** *(unless otherwise indicated, for every kind of regional protection available)*: ARIPO (BW, GH, GM, KE, LR, LS, MW, MZ, NA, RW, SD, SL, ST, SZ, TZ, UG, ZM, ZW), Eurasian (AM, AZ, BY, KG, KZ, RU, TJ, TM), European (AL, AT, BE, BG, CH, CY, CZ, DE, DK, EE, ES, FI, FR, GB, GR, HR, HU, IE, IS, IT, LT, LU, LV, MC, MK, MT, NL, NO, PL, PT, RO, RS, SE, SI, SK, SM, TR), OAPI (BF, BJ, CF, CG, CI, CM, GA, GN, GQ, GW, KM, ML, MR, NE, SN, TD, TG).

[Continued on next page]

(54) **Title**: STARTING APPLICATION PROCESSORS OF A VIRTUAL MACHINE

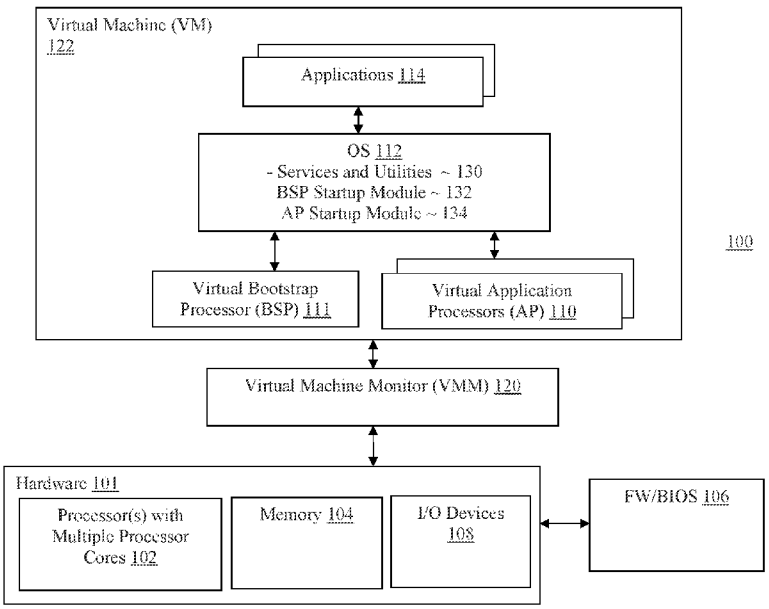


Figure 1

(57) **Abstract**: Apparatuses, methods and storage medium associated with virtual machine application processor startup, are disclosed herein. In embodiments, an apparatus for computing may include a plurality of processor cores; and a plurality of OS modules of an OS. The OS modules may include a BSP module and an AP module. The BSP module may be configured to write into a storage area a start state of an AP of a VM, while the VM is being started up; and the AP module may be configured to start the AP at the start state, directly in a protected mode of execution without first going through a real mode of execution. Other embodiments may be described and/or claimed.

WO 2017/112126 A1

Published:

— *with international search report (Art. 21(3))*

STARTING APPLICATION PROCESSORS OF A VIRTUAL MACHINE

Related Application

This application claims priority to U.S. Patent Application 14/976,990, entitled "STARTING APPLICATION PROCESSORS OF A VIRTUAL MACHINE," filed December 21, 2015.

Technical Field

The present disclosure relates to the field of computing. More particularly, the present disclosure relates to the starting of application processors of a virtual machine.

Background

The background description provided herein is for the purpose of generally presenting the context of the disclosure. Unless otherwise indicated herein, the materials described in this section are not prior art to the claims in this application and are not admitted to be prior art by inclusion in this section.

Typically, during the startup of a virtual machine (VM), the guest operating system (OS) of the VM first boots the virtual Boot Strapping Processor (BSP) of the VM, which in turn, sequentially boots all virtual Application Processors (AP) of the VM in a specified order. For Intel x86 execution environment, the virtual BSP (hereinafter, simply BSP) sends an INIT-SIPI-SIPI IPI sequence to start the virtual AP (hereinafter, simply AP). Each woken AP will begin execution in x86 real mode, and later switches into a protected mode, e.g., a 32-bit protected mode. Accordingly, the virtual machine monitor (VMM) (also referred to as hypervisor) is required to support real mode execution during AP startup, which increases the complexity of the VMM. (INIT = Initialization, IPI = Inter-Processor Interrupt, SIPI = Startup IPI.)

Brief Description of the Drawings

Embodiments will be readily understood by the following detailed description in conjunction with the accompanying drawings. To facilitate this description, like reference numerals designate like structural elements. Embodiments are illustrated by way of example, and not by way of limitation, in the figures of the accompanying drawings.

Figure 1 illustrates a computing device having the virtual machine application processor startup technology of the present disclosure, according to various embodiments.

Figure 2 illustrates an example process of the BSP startup module, according to

various embodiments.

Figure 3 illustrates an example process of the VMM, according to the various embodiments.

Figure 4 illustrates an example process of the AP startup module, according to various embodiments.

Figure 5 illustrates an example computer system suitable for use to practice aspects of the present disclosure, according to various embodiments.

Figure 6 illustrates a storage medium having instructions for practicing methods described with references to Figures 2-4, according to various embodiments.

Detailed Description

Apparatuses, methods and storage medium associated with virtual machine application processor startup, are disclosed herein. In embodiments, an apparatus for computing may include a plurality of processor cores; and a plurality of operating system modules of an operating system. The operating system modules may include a BSP startup module to be operated by a first of the plurality of processor cores selected to be a BSP of a VM while the VM is being started up, to write into a storage area a start state of an AP of the VM; and an AP startup module to be operated by a second of the plurality of processor cores selected to be the AP, to start the AP at the start state, directly in a protected mode of execution without first going through a real mode of execution.

In the following detailed description, reference is made to the accompanying drawings which form a part hereof wherein like numerals designate like parts throughout, and in which is shown by way of illustration embodiments that may be practiced. It is to be understood that other embodiments may be utilized and structural or logical changes may be made without departing from the scope of the present disclosure. Therefore, the following detailed description is not to be taken in a limiting sense, and the scope of embodiments is defined by the appended claims and their equivalents.

Aspects of the disclosure are disclosed in the accompanying description. Alternate embodiments of the present disclosure and their equivalents may be devised without parting from the spirit or scope of the present disclosure. It should be noted that like elements disclosed below are indicated by like reference numbers in the drawings.

Various operations may be described as multiple discrete actions or operations in turn, in a manner that is most helpful in understanding the claimed subject matter. However, the order of description should not be construed as to imply that these operations are necessarily order dependent. In particular, these operations may not be

performed in the order of presentation. Operations described may be performed in a different order than the described embodiment. Various additional operations may be performed and/or described operations may be omitted in additional embodiments.

For the purposes of the present disclosure, the phrase "A and/or B" means (A), (B), or (A and B). For the purposes of the present disclosure, the phrase "A, B, and/or C" means (A), (B), (C), (A and B), (A and C), (B and C), or (A, B and C).

The description may use the phrases "in an embodiment," or "in embodiments," which may each refer to one or more of the same or different embodiments. Furthermore, the terms "comprising," "including," "having," and the like, as used with respect to embodiments of the present disclosure, are synonymous.

As used herein, the term "module" may refer to, be part of, or include an Application Specific Integrated Circuit (ASIC), an electronic circuit, a processor (shared, dedicated, or group) and/or memory (shared, dedicated, or group) that execute one or more software or firmware programs, a combinational logic circuit, and/or other suitable components that provide the described functionality.

Referring now to Figure 1, wherein a computing device having the virtual machine application processor startup technology of the present disclosure, according to various embodiments, is shown. As illustrated, in embodiments, computing device 100 may include hardware 101, firmware (FW)/basic input/output services (BIOS) 106, VMM 120 and one or more VMs 122, operatively coupled with each other as shown. Hardware 101 may include one or more processors 102 with multiple processor cores. VM 122 may include a virtual BSP 111 mapped to a selected one of the processors cores of processor 102, and a number of virtual APs 110 mapped to selected corresponding ones of the processor cores of processor 102. VM 122 may further include OS 112 and a number of applications 114. OS 112 may include a number of services and/or utilities 130, which may include a BSP startup module 132 and an AP startup module 134. Applications 114 may be any application known in the art. As will be described in more detail below, BSP startup module 132, AP startup module 134 and VMM 130 may be incorporated with the virtual machine AP startup technology of the present disclosure to directly start APs 110 in a protected mode of execution without first going through a real mode of execution. These and other aspects related to application execution enclave management will be further described below with references to Figures 2-4.

Processor(s) 102 may be any one of a number of processors known in the art, having one or more processor cores. In embodiments, hardware 101 may further include

memory 104, I/O devices 108, or other elements (not shown). Memory 104 may be any known volatile or non-volatile memory in the art, suitable for storing data. Memory 104 may include a hierarchy of cache memory and system memory. Both the cache and system memory may be respectively organized into cache pages and memory pages. Examples of I/O devices 108 may include communication or networking interfaces, such as Ethernet, WiFi, 3G/4G, Bluetooth®, Near Field Communication, Universal Serial Bus (USB) and so forth, storage devices, such as solid state, magnetic and/or optical drives, input devices, such as keyboard, mouse, touch sensitive screen, and so forth, and output devices, such as, display devices, printers, and so forth.

FW/BIOS 106 may be any one of a number FW/BIOS known in the art. Except for virtual machine application processor startup technology, OS 112 may likewise be any one of a number of OS known in the art, e.g., the Windows OS from Microsoft® Corporation. Applications 114 may likewise be any one of a number of applications known in the art.

Referring now to Figure 2, wherein an example process of the BSP module, according to various embodiments, is shown. As illustrated, process 200 may include the operations performed at blocks 202-208. The operations at blocks 202-208 may be performed e.g., by BSP module 132 of Figure 1, operated by virtual BSP 111 mapped to a selected one of the processor cores of processor 102.

Process 200 may start at block 202. At block 202, a determination may be made on whether there are still more AP to be started for a VM being started. The determination may be performed e.g., by BSP module 132 on bootup by OS 112 at startup of VM 122.

If there are still more AP to be started, at block 204, the startup state of the next AP to be started may be set up. In embodiments, the startup state of the next AP to be started may be set up in a general purpose register of the BSP of the VM. The startup state may include various execution control values at startup, e.g., the next instruction pointer, stack pointer values, memory parameters, and so forth. In embodiments, e.g., in X86 execution environments, the startup state may include the value of the (extended) instruction pointer (IP or EIP) register, the values of the various segmentation registers (i.e., the global descriptor table (GDT), and the CS, DS, ES, FS registers), and so forth. In embodiments, the control register may be the model specific register (MSR) of the BSP of the VM.

Next, at block 206, on setting up of the startup state of the next AP to be started, execution control may be transferred to the VMM. The transfer of the execution control may explicitly or implicitly include the location of the data of the startup state. In

embodiments, writing into the control register (e.g., the MSR) of the BSP may result in a trap from the control register to the VMM.

Next at block 208, process 200 may wait for a startup completion signal or message from the AP being started. Eventually, on receipt of the startup completion signal or message from the AP being started, process 200 may return to block 202, and continue therefrom, as earlier described.

Eventually, when all APs of the VM being started has been started, from block 202, process 200 may end (as opposed to proceed to block 204).

Figure 3 illustrates an example process of the VMM, according to the various embodiments. As illustrated, process 300 may include the operations performed at blocks 302-306. The operations at blocks 302-306 may be performed e.g., by VMM 120 of Figure 1, operated by one or more of the processor cores of processor 102.

Process 300 may start at block 302. At block 302, the startup state of the next AP to be started for a VM being started (as set up by the BSP module of the VM) may be retrieved. For example, on receipt of execution control (as trapped from the control register of the VM), VMM 120 may retrieve the startup state of the next AP to be started from the general purpose register of the corresponding BSP (where the trap came from).

Next, at block 304, the retrieved startup state of the next AP to be started may be programmed into a control structure of the VM, accessible to the AP to be started.

Next, at block 306, the AP may be started (e.g., by the VMM) to execute at the start state programmed into the control structure of the VM.

Figure 4 illustrates an example process of the application processor startup module, according to various embodiments. As illustrated, process 400 may include the operations performed at blocks 402-406. The operations at blocks 402-406 may be performed e.g., by AP module 134 of Figure 1, operated by one of the virtual APs 110 mapped to a selected one of the processor cores of processor 102.

Process 400 may start at block 402. At 402, execution of AP may start directly at a protected mode, without first going through a real mode of execution, at the start state as programmed in the control structure of the VM. In embodiments, the protected mode of execution may be started with memory paging.

At block 404, on retrieval of the start state as programmed in the control structure of the VM, the AP may be initialized.

At block 406, on completion of initialization, the BSP may be signaled to notify the BSP of initialization complete, or a completion message may be sent.

Accordingly, an improved approach to starting APs of a VM has been described. The advantage of the virtual machine AP startup technology of the present disclosure may include speeding up the virtual machine AP startup process, and/or reduces the complexity of the VMM, by not requiring the VMM to support real-mode execution during AP startup. The advantage may be significant, especially when a VM is configured with multiple APs to be started.

Figure 5 illustrates an example computer system that may be suitable for use to practice selected aspects of the present disclosure. As shown, computer 500 may include one or more processors with multiple processor cores 502, read-only memory (ROM) 503, and system memory 504. Additionally, computer system 500 may include mass storage devices 506. Example of mass storage devices 506 may include, but are not limited to, tape drives, hard drives, compact disc read-only memory (CD-ROM) and so forth). Further, computer system 500 may include input/output devices 508 (such as display, keyboard, cursor control and so forth) and communication interfaces 510 (such as network interface cards, modems and so forth). The elements may be coupled to each other via system bus 512, which may represent one or more buses. In the case of multiple buses, they may be bridged by one or more bus bridges (not shown).

Each of these elements may perform its conventional functions known in the art. In particular, ROM 503 may include basic input/output system services (BIOS) 505. System memory 504 and mass storage devices 506 may be employed to store a working copy and a permanent copy of the programming instructions implementing the operations associated with BSP startup module 132, AP startup module 134, and/or VMM 120 as earlier described, collectively referred to as computational logic 522. The various elements may be implemented by assembler instructions supported by processor(s) 502 or high-level languages, such as, for example, C, that can be compiled into such instructions.

The number, capability and/or capacity of these elements 510 - 512 may vary, depending on whether computer system 500 is used as a mobile device, such as a wearable device, a smartphone, a computer tablet, a laptop and so forth, or a stationary device, such as a desktop computer, a server, a game console, a set-top box, an infotainment console, and so forth. Otherwise, the constitutions of elements 510-512 are known, and accordingly will not be further described.

As will be appreciated by one skilled in the art, the present disclosure may be embodied as methods or computer program products. Accordingly, the present disclosure, in addition to being embodied in hardware as earlier described, may take the form of an

entirely software embodiment (including firmware, resident software, micro-code, etc.) or an embodiment combining software and hardware aspects that may all generally be referred to as a "circuit," "module" or "system." Furthermore, the present disclosure may take the form of a computer program product embodied in any tangible or non-transitory medium of expression having computer-usable program code embodied in the medium. Figure 6 illustrates an example computer-readable non-transitory storage medium that may be suitable for use to store instructions that cause an apparatus, in response to execution of the instructions by the apparatus, to practice selected aspects of the present disclosure. As shown, non-transitory computer-readable storage medium 602 may include a number of programming instructions 604. Programming instructions 604 may be configured to enable a device, e.g., computer 500, in response to execution of the programming instructions, to implement (aspects of) BSP startup module 132, AP startup module 134 and/or VMM 120. In alternate embodiments, programming instructions 604 may be disposed on multiple computer-readable non-transitory storage media 602 instead. In still other embodiments, programming instructions 604 may be disposed on computer-readable transitory storage media 602, such as, signals.

Any combination of one or more computer usable or computer readable medium(s) may be utilized. The computer-usable or computer-readable medium may be, for example but not limited to, an electronic, magnetic, optical, electromagnetic, infrared, or semiconductor system, apparatus, device, or propagation medium. More specific examples (a non- exhaustive list) of the computer-readable medium would include the following: an electrical connection having one or more wires, a portable computer diskette, a hard disk, a random access memory (RAM), a read-only memory (ROM), an erasable programmable read-only memory (EPROM or Flash memory), an optical fiber, a portable compact disc read-only memory (CD-ROM), an optical storage device, a transmission media such as those supporting the Internet or an intranet, or a magnetic storage device. Note that the computer- usable or computer-readable medium could even be paper or another suitable medium upon which the program is printed, as the program can be electronically captured, via, for instance, optical scanning of the paper or other medium, then compiled, interpreted, or otherwise processed in a suitable manner, if necessary, and then stored in a computer memory. In the context of this document, a computer-usable or computer-readable medium may be any medium that can contain, store, communicate, propagate, or transport the program for use by or in connection with the instruction execution system, apparatus, or device. The computer- usable medium may include a propagated data signal

with the computer-usable program code embodied therewith, either in baseband or as part of a carrier wave. The computer usable program code may be transmitted using any appropriate medium, including but not limited to wireless, wireline, optical fiber cable, RF, etc.

Computer program code for carrying out operations of the present disclosure may be written in any combination of one or more programming languages, including an object oriented programming language such as Java, Smalltalk, C++ or the like and conventional procedural programming languages, such as the "C" programming language or similar programming languages. The program code may execute entirely on the user's computer, partly on the user's computer, as a stand-alone software package, partly on the user's computer and partly on a remote computer or entirely on the remote computer or server. In the latter scenario, the remote computer may be connected to the user's computer through any type of network, including a local area network (LAN) or a wide area network (WAN), or the connection may be made to an external computer (for example, through the Internet using an Internet Service Provider).

The present disclosure is described with reference to flowchart illustrations and/or block diagrams of methods, apparatus (systems) and computer program products according to embodiments of the disclosure. It will be understood that each block of the flowchart illustrations and/or block diagrams, and combinations of blocks in the flowchart illustrations and/or block diagrams, can be implemented by computer program instructions. These computer program instructions may be provided to a processor of a general purpose computer, special purpose computer, or other programmable data processing apparatus to produce a machine, such that the instructions, which execute via the processor of the computer or other programmable data processing apparatus, create means for implementing the functions/acts specified in the flowchart and/or block diagram block or blocks.

These computer program instructions may also be stored in a computer-readable medium that can direct a computer or other programmable data processing apparatus to function in a particular manner, such that the instructions stored in the computer-readable medium produce an article of manufacture including instruction means which implement the function/act specified in the flowchart and/or block diagram block or blocks.

The computer program instructions may also be loaded onto a computer or other programmable data processing apparatus to cause a series of operational steps to be performed on the computer or other programmable apparatus to produce a computer

implemented process such that the instructions which execute on the computer or other programmable apparatus provide processes for implementing the functions/acts specified in the flowchart and/or block diagram block or blocks.

The flowchart and block diagrams in the figures illustrate the architecture, functionality, and operation of possible implementations of systems, methods and computer program products according to various embodiments of the present disclosure. In this regard, each block in the flowchart or block diagrams may represent a module, segment, or portion of code, which comprises one or more executable instructions for implementing the specified logical function(s). It should also be noted that, in some alternative implementations, the functions noted in the block may occur out of the order noted in the figures. For example, two blocks shown in succession may, in fact, be executed substantially concurrently, or the blocks may sometimes be executed in the reverse order, depending upon the functionality involved. It will also be noted that each block of the block diagrams and/or flowchart illustration, and combinations of blocks in the block diagrams and/or flowchart illustration, can be implemented by special purpose hardware-based systems that perform the specified functions or acts, or combinations of special purpose hardware and computer instructions.

The terminology used herein is for the purpose of describing particular embodiments only and is not intended to be limiting of the disclosure. As used herein, the singular forms “a,” “an” and “the” are intended to include plural forms as well, unless the context clearly indicates otherwise. It will be further understood that the terms “comprises” and/or “comprising,” when used in this specification, specific the presence of stated features, integers, steps, operations, elements, and/or components, but do not preclude the presence or addition of one or more other features, integers, steps, operation, elements, components, and/or groups thereof.

Embodiments may be implemented as a computer process, a computing system or as an article of manufacture such as a computer program product of computer readable media. The computer program product may be a computer storage medium readable by a computer system and encoding a computer program instructions for executing a computer process.

The corresponding structures, material, acts, and equivalents of all means or steps plus function elements in the claims below are intended to include any structure, material or act for performing the function in combination with other claimed elements are specifically claimed. The description of the present disclosure has been presented for

purposes of illustration and description, but is not intended to be exhaustive or limited to the disclosure in the form disclosed. Many modifications and variations will be apparent to those of ordinary skill without departing from the scope and spirit of the disclosure. The embodiment was chosen and described in order to best explain the principles of the disclosure and the practical application, and to enable others of ordinary skill in the art to understand the disclosure for embodiments with various modifications as are suited to the particular use contemplated.

Referring back to Figure 5, for one embodiment, at least one of processors 602 may be packaged together with memory having aspects of BSP startup module 132, AP startup module 134 and/or VMM 120. For one embodiment, at least one of processors 602 may be packaged together with memory having aspects of BSP startup module 132, AP startup module 134 and/or VMM 120, to form a System in Package (SiP). For one embodiment, at least one of processors 602 may be integrated on the same die with memory having aspects of BSP startup module 132, AP startup module 134 and/or VMM 120. For one embodiment, at least one of processors 602 may be packaged together with memory having aspects of BSP startup module 132, AP startup module 134 and/or VMM 120, to form a System on Chip (SoC). For at least one embodiment, the SoC may be utilized in, e.g., but not limited to, a wearable device, a smartphone or a computing tablet.

Thus various example embodiments of the present disclosure have been described including, but are not limited to:

Example 1 may be an apparatus for computing, comprising: a plurality of processor cores; and a plurality of operating system modules of an operating system, including: a bootstrap processor startup module to be operated by a first of the plurality of processor cores selected to be a bootstrap processor of a virtual machine while the virtual machine is being started up, to write into a storage area a start state of an application processor of the virtual machine; and an application processor startup module to be operated by a second of the plurality of processor cores selected to be the application processor, to start the application processor at the start state, directly in a protected mode of execution without first going through a real mode of execution.

Example 2 may be example 1, wherein the bootstrap processor startup module, on writing the start state of the application processor into the storage area, may further transfer execution control to a virtual machine manager of the apparatus.

Example 3 may be example 2, wherein the transfer of execution control may include explicit or implicit provision of the location of the storage area to the virtual

machine manager.

Example 4 may be example 2, wherein the storage area may be a control register of the bootstrap processor, and the execution control may be transferred to the virtual machine manager via a trap from the control register.

Example 5 may be any one of examples 2-4, further comprising the virtual machine manager, wherein the virtual machine manager may be operated by the one or more processor cores to retrieve the start state of the application from the storage area, program the start state of the application processor into a control structure of the virtual machine accessible to the application processor, and set the application processor to start execution at the start state programmed in the control structure, on receipt of the execution control.

Example 6 may be example 5, wherein the application processor startup module may further start the application processor directly in the protected mode of execution with memory paging.

Example 7 may be example 5, wherein the application processor startup module may further initialize the application processor, and signal the boostrap processor startup module on completion of the initialization.

Example 8 may be example 7, wherein the bootstrap processor startup module may further await for the completion signal prior to perform another write into the storage area of another start state of another application processor of the virtual machine.

Example 9 may be a method for computing, comprising: writing into a storage area of an apparatus, by a bootstrap processor startup module operated by a first of a plurality of processor cores of the apparatus selected to be a bootstrap processor of a virtual machine, a start state of an application processor of the virtual machine, while the virtual machine may be being started up; and starting, by an application processor startup module to be operated by a second of the plurality of processor cores selected to be the application processor, the application processor at the start state, directly in a protected mode of execution without first going through a real mode of execution.

Example 10 may be example 9, further comprising transferring execution control, by the bootstrap processor startup module, to a virtual machine manager of the apparatus, on writing the start state of the application processor into the storage area.

Example 11 may be example 10, wherein transferring execution control may include explicitly or implicitly providing the location of the storage area to the virtual machine manager.

Example 12 may be example 10, wherein the storage area may be a control register of the bootstrap processor, and transferring of the execution control comprises trapping to the virtual machine manager from the control register.

Example 13 may be any one of examples 10-12, further comprising retrieving, by the virtual machine manager, the start state of the application from the storage area, programming the start state of the application processor into a control structure of the virtual machine accessible to the application processor, and setting the application processor to start execution at the start state programmed in the control structure, on receipt of the execution control.

Example 14 may be example 13, further comprising starting, by the application processor startup module, the application processor directly in the protected mode of execution with memory paging.

Example 15 may be example 13, further comprising initializing, by the application processor startup module, the application processor, and signaling the boostrap processor startup module on completion of the initialization.

Example 16 may be example 15, further comprising awaiting, by the bootstrap processor startup module, the completion signal prior to perform another write into the storage area of another start state of another application processor of the virtual machine.

Example 17 may be one or more computer-readable media comprising instructions that cause a computing device, in response to execution of the instructions by a plurality of processor cores of the computing device, to provide: a bootstrap processor startup module to be operated by a first of the plurality of processor cores selected to be a bootstrap processor of a virtual machine while the virtual machine may be being started up, to write into a storage area of the computing device a start state of an application processor of the virtual machine; and an application processor startup module to be operated by a second of the plurality of processor cores selected to be the application processor, to start the application processor at the start state, directly in a protected mode of execution without first going through a real mode of execution.

Example 18 may be example 17, wherein the bootstrap processor startup module, on writing the start state of the application processor into the storage area, may further transfer execution control to a virtual machine manager of the apparatus.

Example 19 may be example 18, wherein the transfer of execution control may include explicit or implicit provision of the location of the storage area to the virtual machine manager.

Example 20 may be example 18, wherein the storage area may be a control register of the bootstrap processor, and the execution control may be transferred to the virtual machine manager via a trap from the control register.

Example 21 may be any one of examples 18-20, further comprising the virtual machine manager, wherein the virtual machine manager may be operated by the one or more processor cores to retrieve the start state of the application from the storage area, program the start state of the application processor into a control structure of the virtual machine accessible to the application processor, and set the application processor to start execution at the start state programmed in the control structure, on receipt of the execution control.

Example 22 may be example 21, wherein the application processor startup module may further start the application processor directly in the protected mode of execution with memory paging.

Example 23 may be example 21, wherein the application processor startup module may further initialize the application processor, and signal the boostrap processor startup module on completion of the initialization.

Example 24 may be example 23, wherein the bootstrap processor startup module may further await for the completion signal prior to perform another write into the storage area of another start state of another application processor of the virtual machine.

Example 25 may be an apparatus for computing, comprising: bootstrap processor means for writing into a storage area of a bootstrap processor of a virtual machine of the apparatus, a start state of an application processor of the virtual machine, while the virtual machine may be being started up; and application processor means for starting the application processor at the start state, directly in a protected mode of execution without first going through a real mode of execution.

Example 26 may be example 25, wherein the bootstrap processor means may comprise means for transferring execution control to a virtual machine manager of the apparatus, on writing the start state of the application processor into the storage area.

Example 27 may be example 26, wherein transferring execution control may include explicitly or implicitly providing the location of the storage area to the virtual machine manager.

Example 28 may be example 26, wherein the storage area may be a control register of the bootstrap processor, and transferring of the execution control comprises trapping to the virtual machine manager from the control register.

Example 29 may be example 26-28, wherein the virtual machine manager may comprise means for retrieving the start state of the application from the storage area, means for programming the start state of the application processor into a control structure of the virtual machine accessible to the application processor, and means for setting the application processor to start execution at the start state programmed in the control structure, on receipt of the execution control.

Example 30 may be example 29, wherein the application processor means comprises means for starting the application processor directly in the protected mode of execution with memory paging.

Example 31 may be example 29, the application processor means may comprise means for initializing the application processor, and means for signaling the boostrap processor startup module on completion of the initialization.

Example 32 may be example 31, wherein the bootstrap processor means may comprise means for awaiting the completion signal prior to perform another write into the storage area of another start state of another application processor of the virtual machine.

It will be apparent to those skilled in the art that various modifications and variations can be made in the disclosed embodiments of the disclosed device and associated methods without departing from the spirit or scope of the disclosure. Thus, it is intended that the present disclosure covers the modifications and variations of the embodiments disclosed above provided that the modifications and variations come within the scope of any claims and their equivalents.

Claims

What is claimed is:

1. An apparatus for computing, comprising:

a plurality of processor cores; and

a plurality of operating system modules of an operating system, including:

a bootstrap processor startup module to be operated by a first of the plurality of processor cores selected to be a bootstrap processor of a virtual machine while the virtual machine is being started up, to write into a storage area a start state of an application processor of the virtual machine; and

an application processor startup module to be operated by a second of the plurality of processor cores selected to be the application processor, to start the application processor at the start state, directly in a protected mode of execution without first going through a real mode of execution.

2. The apparatus of claim 1, wherein the bootstrap processor startup module, on writing the start state of the application processor into the storage area, is to further transfer execution control to a virtual machine manager of the apparatus.

3. The apparatus of claim 2, wherein the transfer of execution control includes explicit or implicit provision of the location of the storage area to the virtual machine manager.

4. The apparatus of claim 2, wherein the storage area is a control register of the bootstrap processor, and the execution control is transferred to the virtual machine manager via a trap from the control register.

5. The apparatus of any one of claims 2-4, further comprising the virtual machine manager, wherein the virtual machine manager is to be operated by the one or more processor cores to retrieve the start state of the application from the storage area, program the start state of the application processor into a control structure of the virtual machine accessible to the application processor, and set the application processor to start execution at the start state programmed in the control structure, on receipt of the execution control.

6. The apparatus of claim 5, wherein the application processor startup module is to further start the application processor directly in the protected mode of execution with memory paging.

7. The apparatus of claim 5, wherein the application processor startup module is to further initialize the application processor, and signal the boostrap processor startup module on completion of the initialization.

8. The apparatus of claim 7, wherein the bootstrap processor startup module is to further await for the completion signal prior to perform another write into the storage area of another start state of another application processor of the virtual machine.

9. A method for computing, comprising:

writing into a storage area of an apparatus, by a bootstrap processor startup module operated by a first of a plurality of processor cores of the apparatus selected to be a bootstrap processor of a virtual machine, a start state of an application processor of the virtual machine, while the virtual machine is being started up; and

starting, by an application processor startup module to be operated by a second of the plurality of processor cores selected to be the application processor, the application processor at the start state, directly in a protected mode of execution without first going through a real mode of execution.

10. The method of claim 9, further comprising transferring execution control, by the bootstrap processor startup module, to a virtual machine manager of the apparatus, on writing the start state of the application processor into the storage area.

11. The method of claim 10, wherein transferring execution control includes explicitly or implicitly providing the location of the storage area to the virtual machine manager.

12. The method of claim 10, wherein the storage area is a control register of the bootstrap processor, and transferring of the execution control comprises trapping to the virtual machine manager from the control register.

13. The method of claim 10, further comprising retrieving, by the virtual machine manager, the start state of the application from the storage area, programming the start state of the application processor into a control structure of the virtual machine accessible to the application processor, and setting the application processor to start execution at the start state programmed in the control structure, on receipt of the execution control.

14. The method of claim 13, further comprising starting, by the application processor startup module, the application processor directly in the protected mode of execution with memory paging.

15. The method of claim 13, further comprising initializing, by the application processor startup module, the application processor, and signaling the boostrap processor startup module on completion of the initialization.

16. The method of claim 15, further comprising awaiting, by the bootstrap processor startup module, the completion signal prior to perform another write into the storage area of another start state of another application processor of the virtual machine.

17. One or more computer-readable media comprising instructions that cause a computing device, in response to execution of the instructions by a plurality of processor cores of the computing device, to practice any one of the methods of claims 9 – 16.

18. An apparatus for computing, comprising:

bootstrap processor means for writing into a storage area of a bootstrap processor of a virtual machine of the apparatus, a start state of an application processor of the virtual machine, while the virtual machine is being started up; and

application processor means for starting the application processor at the start state, directly in a protected mode of execution without first going through a real mode of execution.

19. The apparatus of claim 18, wherein the bootstrap processor means comprises means for transferring execution control to a virtual machine manager of the apparatus, on writing the start state of the application processor into the storage area.

20. The apparatus of claim 19, wherein transferring execution control includes explicitly or implicitly providing the location of the storage area to the virtual machine manager.

21. The apparatus of claim 19, wherein the storage area is a control register of the bootstrap processor, and transferring of the execution control comprises trapping to the virtual machine manager from the control register.

22. The apparatus of any one of claims 19-21, wherein the virtual machine manager comprises means for retrieving the start state of the application from the storage area, means for programming the start state of the application processor into a control structure of the virtual machine accessible to the application processor, and means for setting the application processor to start execution at the start state programmed in the control structure, on receipt of the execution control.

23. The apparatus of claim 22, wherein the application processor means comprises means for starting the application processor directly in the protected mode of execution with memory paging.

24. The apparatus of claim 22, the application processor means comprises means for initializing the application processor, and means for signaling the boostrap processor startup module on completion of the initialization.

25. The apparatus of claim 24, wherein the bootstrap processor means comprises means for awaiting the completion signal prior to perform another write into the storage area of another start state of another application processor of the virtual machine.

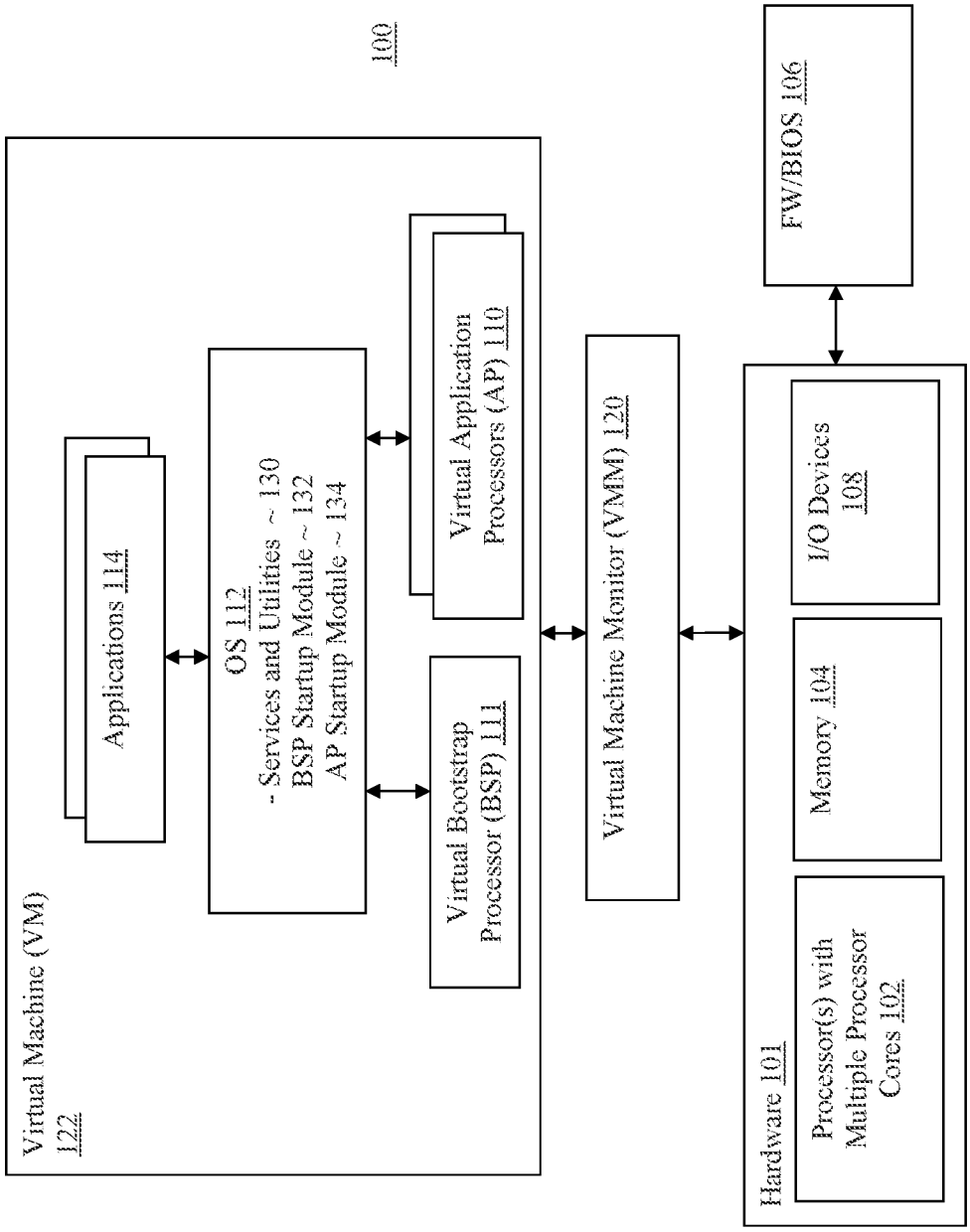

100
Virtual Machine (VM)
122
Applications 114
OS 112
- Services and Utilities ~ 130
BSP Startup Module ~ 132
AP Startup Module ~ 134
Virtual Application Processors (AP) 110
Virtual Bootstrap Processor (BSP) 111
Virtual Machine Monitor (VMM) 120
FW/BIOS 106
Hardware 101
Processor(s) with Multiple Processor Cores 102
Memory 104
I/O Devices 108

Figure 1

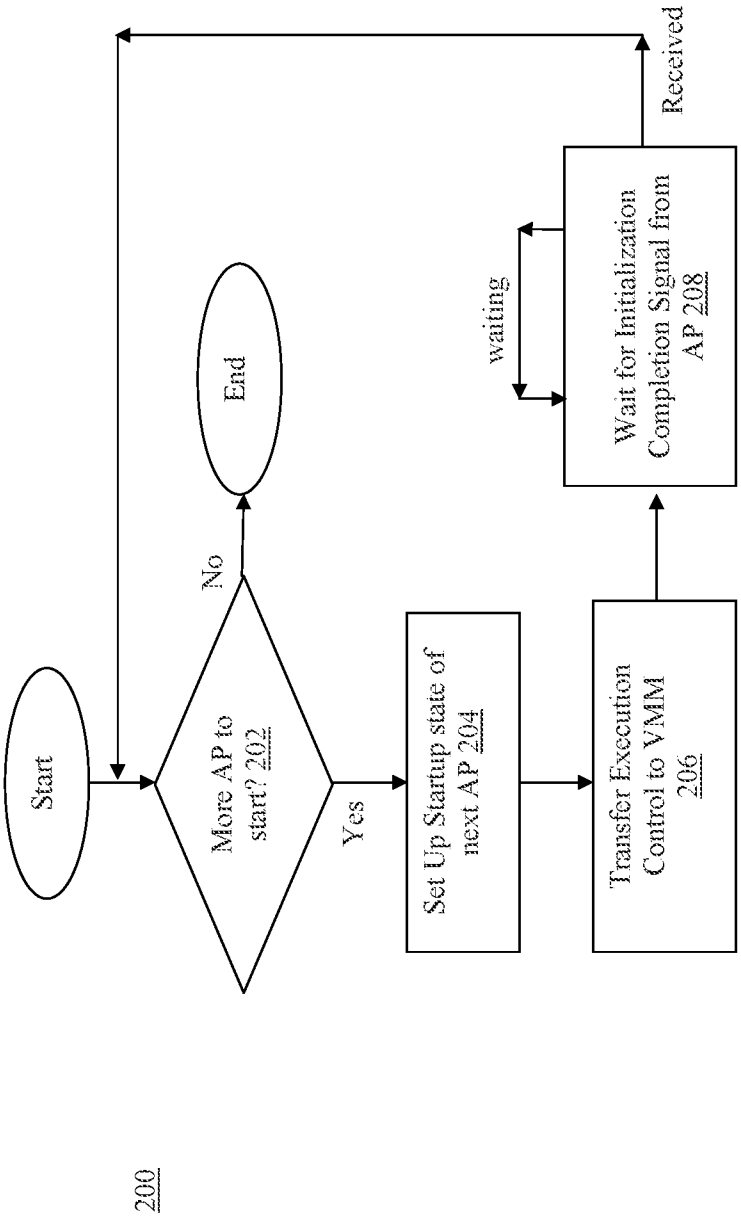

Start
More AP to start? 202
No
End
Yes
Set Up Startup state of next AP 204
Transfer Execution Control to VMM 206
Wait for Initialization Completion Signal from AP 208
waiting
Received
200

Figure 2

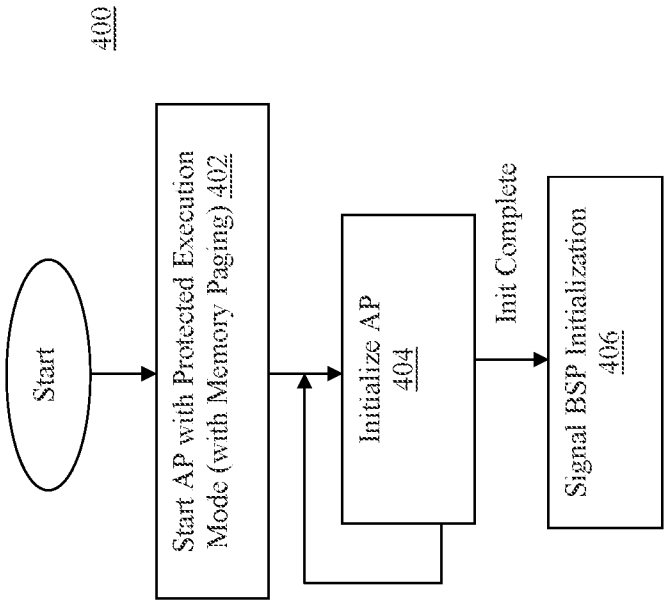


Figure 4

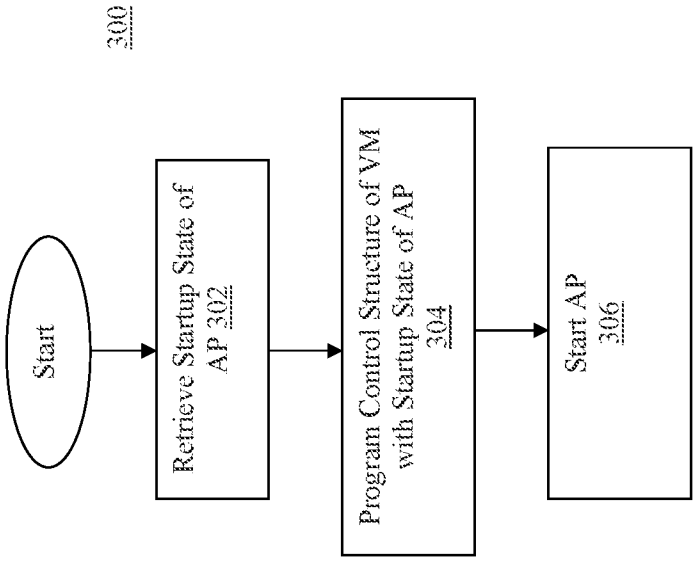


Figure 3

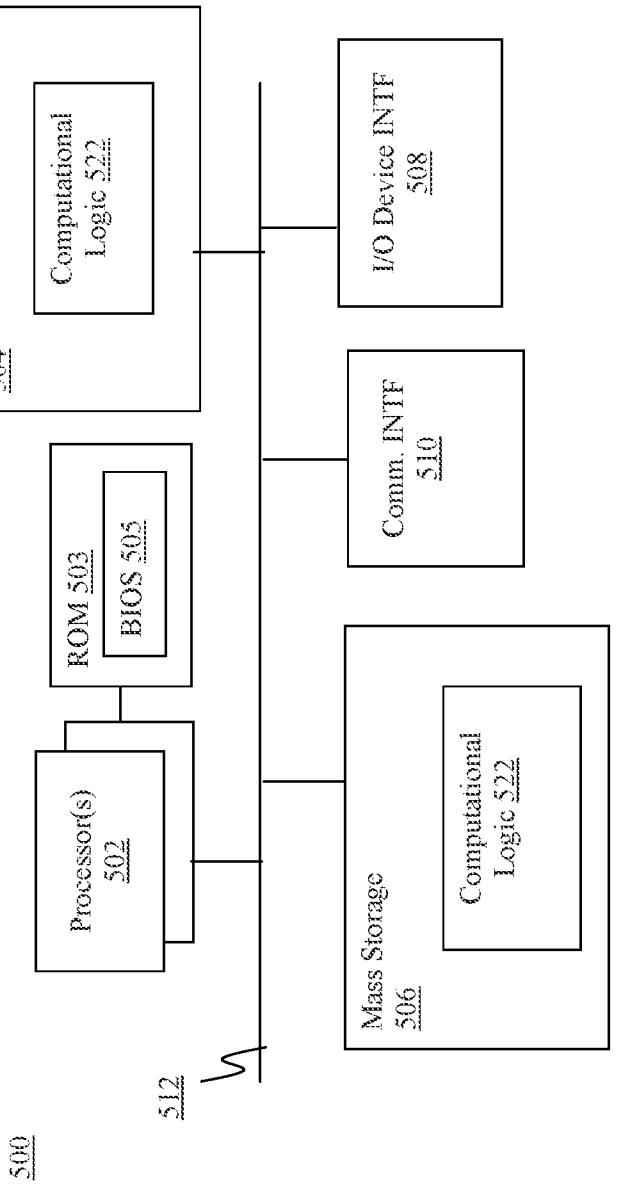

500
512
Processor(s)
502
ROM 503
BIOS 505
Memory
504
Computational
Logic 522
Mass Storage
506
Computational
Logic 522
Comm. INTF
510
I/O Device INTF
508

Figure 5

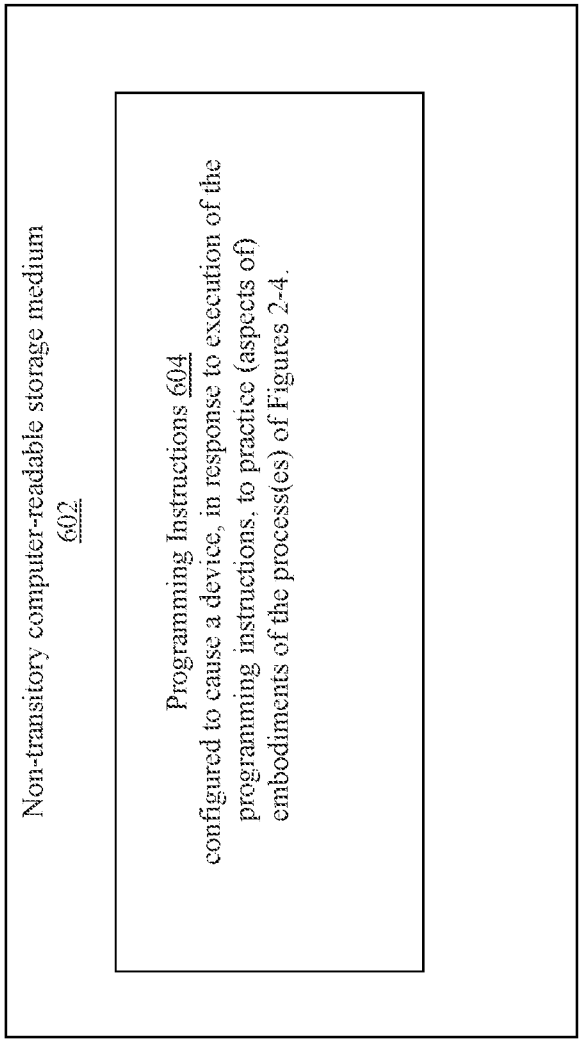

Non-transitory computer-readable storage medium
602
Programming Instructions 604
configured to cause a device, in response to execution of the programming instructions, to practice (aspects of) embodiments of the process(es) of Figures 2-4.

Figure 6

INTERNATIONAL SEARCH REPORT

International application No.
PCT/US2016/061846

A. CLASSIFICATION OF SUBJECT MATTER

G06F 9/44(2006.01)i

According to International Patent Classification (IPC) or to both national classification and IPC

B. FIELDS SEARCHED

Minimum documentation searched (classification system followed by classification symbols)
G06F 9/44; G06F 9/46; H04L 9/00; G06F 9/50; G06F 13/00

Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched
Korean utility models and applications for utility models
Japanese utility models and applications for utility models

Electronic data base consulted during the international search (name of data base and, where practicable, search terms used)
eKOMPASS(KIPO internal) & Keywords: virtual machine, bootstrap processor, application processor, startup, protected mode

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
A	US 2014-0149992 A1 (VINCET J. ZIMMER et al.) 29 May 2014 See paragraphs [0020]-[0051].	1-25
A	US 5255379 A (MICHAEL D. MELO) 19 October 1993 See column 4, line 42 - column 12, line 55; and figure 4.	1-25
A	US 5694606 A (RICHARD A. PLETCHER et al.) 02 December 1997 See column 7, line 22 - column 10, line 31; and figures 9a-10b.	1-25
A	US 2006-0005200 A1 (RENE ANTONIO VEGA et al.) 05 January 2006 See paragraphs [0034]-[0045]; and figures 2-5.	1-25
A	US 2004-0210764 A1 (KEVIN J. MCGRATH et al.) 21 October 2004 See paragraphs [0067]-[0081]; and figure 4.	1-25

☐ Further documents are listed in the continuation of Box C.

☒ See patent family annex.

* Special categories of cited documents:
"A" document defining the general state of the art which is not considered to be of particular relevance
"E" earlier application or patent but published on or after the international filing date
"L" document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified)
"O" document referring to an oral disclosure, use, exhibition or other means
"P" document published prior to the international filing date but later than the priority date claimed
"T" later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention
"X" document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone
"Y" document of particular relevance; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents,such combination being obvious to a person skilled in the art
"&" document member of the same patent family

Date of the actual completion of the international search	Date of mailing of the international search report
27 February 2017 (27.02.2017)	**28 February 2017 (28.02.2017)**

Name and mailing address of the ISA/KR
International Application Division
Korean Intellectual Property Office
189 Cheongsa-ro, Seo-gu, Daejeon, 35208, Republic of Korea
Facsimile No. +82-42-481-8578

Authorized officer
CHIN, Sang Bum
Telephone No. +82-42-481-8398

INTERNATIONAL SEARCH REPORT

Information on patent family members

International application No.
PCT/US2016/061846

Patent document cited in search report	Publication date	Patent family member(s)	Publication date
US 2014-0149992 A1	29/05/2014	CN 101482832 A	15/07/2009
		CN 101482832 B	04/07/2012
		EP 2079019 A1	15/07/2009
		EP 2458502 A1	30/05/2012
		US 2009-172690 A1	02/07/2009
		US 8645965 B2	04/02/2014
US 5255379 A		None	
US 5694606 A	02/12/1997	US 5596755 A	21/01/1997
US 2006-0005200 A1	05/01/2006	CN 100530102 C	19/08/2009
		CN 1716203 A	04/01/2006
		EP 1628215 A2	22/02/2006
		EP 1628215 A3	21/01/2009
		JP 04156611 B2	24/09/2008
		JP 2006-018819 A	19/01/2006
		KR 10-2006-0046262 A	17/05/2006
		US 7260702 B2	21/08/2007
US 2004-0210764 A1	21/10/2004	AU 2003-221972 A1	03/11/2003
		AU 2003-231070 A1	03/11/2003
		CN 100339780 C	26/09/2007
		CN 1285033 C	15/11/2006
		CN 1647011 A	27/07/2005
		CN 1647039 A	27/07/2005
		CN 1647039 C	15/11/2006
		EP 1495394 A2	12/01/2005
		EP 1495394 B1	23/07/2008
		EP 1495401 A2	12/01/2005
		EP 1495401 B1	24/01/2007
		JP 04564756 B2	20/10/2010
		JP 04762494 B2	31/08/2011
		JP 2005-528677 A	22/09/2005
		JP 2005-528678 A	22/09/2005
		KR 10-0921779 B1	15/10/2009
		KR 10-0950102 B1	29/03/2010
		US 2004-0210760 A1	21/10/2004
		US 7603550 B2	13/10/2009
		US 7603551 B2	13/10/2009
		WO 03-090052 A2	30/10/2003
		WO 03-090074 A2	30/10/2003
		WO 2003-090052 A3	12/08/2004
		WO 2003-090074 A3	19/08/2004