



- (51) International Patent Classification:
G06F 12/14 (2006.01) *G06F 12/08* (2006.01)
- (21) International Application Number:
PCT/US2013/047332
- (22) International Filing Date:
24 June 2013 (24.06.2013)
- (25) Filing Language: English
- (26) Publication Language: English
- (30) Priority Data:
13/729,348 28 December 2012 (28.12.2012) US
- (71) Applicant (for all designated States except US): **INTEL CORPORATION** [US/US]; 2200 Mission College Blvd., Santa Clara, California 95054 (US).
- (72) Inventors; and
- (71) Applicants (for US only): **MCKEEN, Francis X.** [US/US]; 10612 NW LeMans Ct., Portland, Oregon 97229 (US). **GOLDSMITH, Michael A.** [US/US]; 17905 Kelok, Lake Oswego, Oregon 97034 (US). **HUNTLEY, Barry E.** [US/US]; 3762 NW 2nd Ave, Hillsboro, Oregon 97124 (US). **JOHNSON, Simon P.** [GB/US]; 14964 SW Opal Drive, Beaverton, Oregon 97007 (US). **LESLIE-HURD, Rebekah** [US/US]; 411 N Holman St., Portland, Oregon

97217 (US). **ROZAS, Carlos V.** [US/US]; 1534 NW Morgan Lane, Portland, Oregon 97229 (US). **SAVAGAONKAR, Uday R.** [IN/US]; 5507 NW 133rd Ave, Portland, Oregon 97229 (US). **SCARLATA, Vincent R.** [US/US]; 17425 SW Lisa Street, Beaverton, Oregon 97006 (US). **SHANBHOGUE, Vedvyas** [IN/US]; 12531 W HWY 71, Apt. P2102, Austin, Texas 78738 (US). **SMITH, Wesley H.** [US/US]; 11309 Brass Kettle Road, Raleigh, North Carolina 27614 (US). **NEIGER, Gilbert** [US/US]; 2111 NE 25th Avenue, JF2-65, Hillsboro, Oregon 97124 (US).

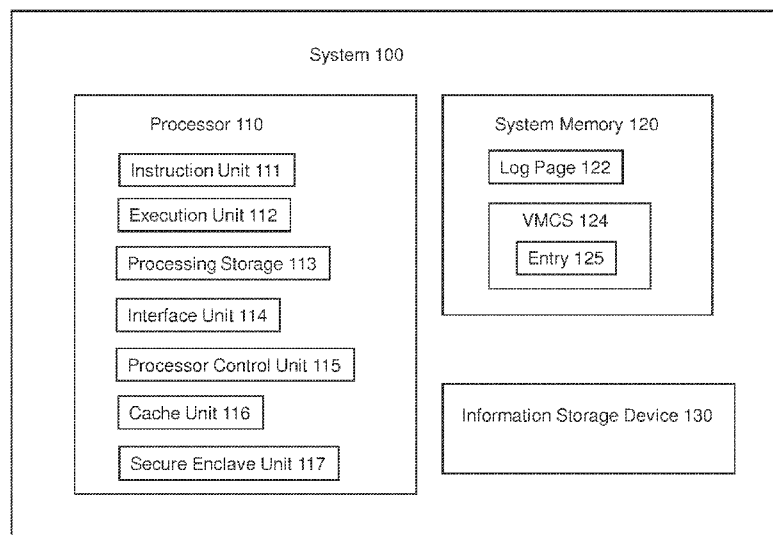
(74) Agent: **LANE, Thomas R.**; CPA Global, P.O. Box 52050, Minneapolis, Minnesota 55402 (US).

(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, DK, DM, DO, DZ, EC, EE, EG, ES, FI, GB, GD, GE, GH, GM, GT, HN, HR, HU, ID, IL, IN, IS, JP, KE, KG, KN, KP, KR, KZ, LA, LC, LK, LR, LS, LT, 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, 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.

[Continued on next page]

(54) Title: LOGGING IN SECURE ENCLAVES

FIGURE 1



(57) Abstract: Embodiments of an invention for logging in secure enclaves are disclosed. In one embodiment, a processor includes an instruction unit and an execution unit. The instruction unit is to receive an instruction having an associated enclave page cache address. The execution unit is to execute the instruction without causing a virtual machine exit, wherein execution of the instruction includes logging the instruction and the associated enclave page cache address.



(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, 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).

Published:

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

LOGGING IN SECURE ENCLAVES

BACKGROUND

1. Field

The present disclosure pertains to the field of information processing, and more particularly, to the field of security in information processing systems.

2. Description of Related Art

Confidential information is stored, transmitted, and used by many information processing systems. Therefore, techniques have been developed to provide for the secure handling and storing of confidential information. These techniques include various approaches to creating and maintaining a secured, protected, or isolated partition or environment within an information processing system.

Brief Description of the Figures

The present invention is illustrated by way of example and not limitation in the accompanying figures.

Figure 1 illustrates a system including logging in secure enclaves according to an embodiment of the present invention.

Figure 2 illustrates a secure enclave unit according to an embodiment of the present invention.

Figure 3 illustrates a method for logging in secure enclaves according to an embodiment of the present invention.

Detailed Description

Embodiments of an invention for logging in secure enclaves are described. In this description, numerous specific details, such as component and system configurations, may be set forth in order to provide a more thorough understanding of the present invention. It will be appreciated, however, by one skilled in the art, that the invention may be practiced without such specific details. Additionally, some well-known structures, circuits, and other features have not been shown in detail, to avoid unnecessarily obscuring the present invention.

In the following description, references to "one embodiment," "an embodiment," "example

embodiment,” “various embodiments,” etc., indicate that the embodiment(s) of the invention so described may include particular features, structures, or characteristics, but more than one embodiment may and not every embodiment necessarily does include the particular features, structures, or characteristics. Further, some embodiments may have some, all, or none of the features described for other embodiments.

As used in the claims, unless otherwise specified the use of the ordinal adjectives “first,” “second,” “third,” etc. to describe an element merely indicate that a particular instance of an element or different instances of like elements are being referred to, and is not intended to imply that the elements so described must be in a particular sequence, either temporally, spatially, in ranking, or in any other manner.

Also, the terms “bits,” “flags,” “fields,” “entries,” etc., may be used to describe any type of storage location in a register, table, database, or other data structure, whether implemented in hardware or software, but are not meant to limit embodiments of the invention to any particular type of storage location or number of bits or other elements within any particular storage location. The term “clear” may be used to indicate storing or otherwise causing the logical value of zero to be stored in a storage location, and the term “set” may be used to indicate storing or otherwise causing the logical value of one, all ones, or some other specified value to be stored in a storage location; however, these terms are not meant to limit embodiments of the present invention to any particular logical convention, as any logical convention may be used within embodiments of the present invention.

As described in the background section, various approaches to creating and maintaining a secured, protected, or isolated partition or environment within an information processing system have been developed. One such approach involves secure enclaves as described in the co-pending U.S. Patent Application entitled “Method and Apparatus to Provide Secure Application Execution,” filed June 19, 2012, Serial No. 13/527,547, which is hereby incorporated by reference as an example of at least one embodiment of a secure enclave. However, the incorporated reference is not intended to limit the scope of embodiments of the invention in any way and other embodiments may be used while remaining within the spirit and scope of the invention.

Figure 1 illustrates system 100, an information processing system including paging in secure enclaves according to an embodiment of the present invention. System 100 may represent any type of information processing system, such as a server, a desktop computer, a portable computer, a set-top box, a hand-held device, or an embedded control system. System 100 includes processor 110, system memory 120, and information storage device 130. Systems

embodying the present invention may include any number of each of these components and any other components or other elements, such as information storage devices, peripherals, and input/output devices. Any or all of the components or other elements in this or any system embodiment, may be connected, coupled, or otherwise in communication with each other
5 through any number of buses, point-to-point, or other wired or wireless interfaces or connections, unless specified otherwise.

System memory 120 may be dynamic random access memory or any other type of medium readable by processor 110. Information storage device 130 may include any type of persistent or non-volatile memory or storage, such as a flash memory and/or a solid state, magnetic, or optical
10 disk drive.

Processor 110 may represent one or more processors integrated on a single substrate or packaged within a single package, each of which may include multiple threads and/or multiple execution cores, in any combination. Each processor represented as processor 110 may be any type of processor, including a general purpose microprocessor, such as a processor in the Intel®
15 Core® Processor Family, Intel® Atom® Processor Family, or other processor family from Intel® Corporation, or another processor from another company, or a special purpose processor or microcontroller. Processor 110 may include instruction unit 111, execution unit 112, processing storage 113, interface unit 114, processor control unit 115, cache unit 116, and secure enclave unit 117. Processor 110 may also include any other circuitry, structures, or logic not
20 shown in Figure 1, and/or any circuitry, structures, or logic shown or described as elsewhere in Figure 1.

Instruction unit 111 may represent any circuitry, structure, or other hardware, such as an instruction decoder, for fetching, receiving, decoding, and/or scheduling instructions. Any instruction format may be used within the scope of the present invention; for example, an
25 instruction may include an opcode and one or more operands, where the opcode may be decoded into one or more micro-instructions or micro-operations for execution by execution unit 112.

Execution unit 112 may include any circuitry, structure, or other hardware, such as an arithmetic unit, logic unit, floating point unit, shifter, etc., for processing data and executing instructions, micro-instructions, and/or micro-operations.

30 Processing storage 113 may represent any type of storage usable for any purpose within processor 110; for example, it may include any number of data registers, instruction registers, status registers, configuration registers, control registers, other programmable or hard-coded registers or register files, or any other storage structures.

Interface unit 114 may represent any circuitry, structure, or other hardware, such as a bus

unit, messaging unit, or any other unit, port, or interface, to allow processor 110 to communicate with other components in system 100 through any type of bus, point to point, or other connection, directly or through any other component, such as a memory controller or a bus bridge.

5 Processor control unit 115 may include any logic, microcode, circuitry, or other hardware to control the operation of the units and other elements of processor 110 and the transfer of data within, into, and out of processor 110. Processor control unit 115 may cause processor 110 to perform or participate in the performance of method embodiments of the present invention, such as the method embodiments described below, for example, by causing processor 110 to execute
10 instructions received by instruction unit 111 and micro-instructions or micro-operations derived from instructions received by instruction unit 111.

Cache unit 116 may represent any one or more levels of cache memory in a memory hierarchy of information processing system 100, implemented in static random access memory or any other memory technology. Cache unit 116 may include any combination of cache
15 memories dedicated to or shared among any one or more execution cores or processors within processor 110 according to any known approaches to caching in information processing systems.

Secure enclave unit 117 may represent any logic, circuitry, hardware, or other structures for creating and maintaining a secured, protected, or isolated environment, such as a secure enclave as described herein, in which an application or other software may run, execute, be
20 loaded, or otherwise be present within an information processing system such as system 100. For purposes of this description, each instance of such an environment may be referred to as a secure enclave, although embodiments of the present invention are not limited to those using a secure enclave as the secured, protected, or isolated environment. In one embodiment, a secure enclave may be created and maintained using instructions in the instruction set of a processor in
25 the Intel® Core® Processor Family or other processor family from Intel® Corporation.

Figure 2 illustrates secure enclave unit 200, an embodiment of which may serve as secure enclave unit 117 in system 100. All or part of secure enclave unit 200 may be included within any one or more other units of processor 110, such as instruction unit 111, execution unit 112, processor storage 113, processor control unit 115, and cache unit 116.

30 Secure enclave unit 200 may include encryption unit 210, which may include any logic, circuitry, or other hardware to execute any one or more encryption algorithms and the corresponding decryption algorithms, and may include logic, circuitry, or other hardware shared with another encryption unit in processor 110.

Secure enclave unit 200 may also include enclave page cache (EPC) 220. In one

embodiment, EPC 220 may be a dedicated portion of cache unit 116, such as a portion of a last level cache. Other embodiments are possible, including embodiments in which all or part of EPC 220 may be outside of processor 110. EPC 220 may be used to store unencrypted code and data for one or more secure enclaves. Access control logic 214, range register(s) 216, and EPC map (EPCM) 218 may be used to prevent access to a page within EPC 220 except by an application running on processor 110 within the secure enclave to which the page is allocated.

Secure enclave unit 200 may also include EPC paging unit 230. EPC paging unit 230 may include any logic, circuitry, or other hardware to provide for moving pages out of and into EPC 220 according to embodiments of the present invention. EPC paging unit 230 may include microcode, logic, circuitry, and/or other hardware to decode and execute an EWB instruction, an ELD instruction, and an EPA instruction.

These instructions may be used by an operating system or other software to manage EPC 220 and provide a virtual memory space for use by one or more secure enclaves that is larger than the size of EPC 220. An EWB instruction may be used to evict a page from EPC 220 to system memory 120. An ELD instruction may be used to load a page from system memory 120 into EPC 220. An EPA instruction may be used to allocate a special EPC page, called a version array page, on which to store version information for evicted pages.

Secure enclave unit 200 may also include EPC memory management unit (MMU) 240. EPC MMU 240 may include microcode, logic, circuitry, and/or other hardware to decode and execute an EADD instruction, an EAUG instruction, an EMOD instruction, an EACCEPT instruction, and an EREMOVE instruction. These instructions may be used by an operating system and/or other software dynamically increase and decrease the size of the memory space in EPC 220 allocated to one or more secure enclaves.

An EADD instruction is a privileged instruction that may be used by an operating system to allocate a page in EPC 220 to a secure enclave. Unlike the allocation of a page using an EAUG instruction, allocation of a page using an EADD instruction does not involve the acceptance of the allocation by the secure enclave using an EACCEPT instruction. An EAUG instruction is a privileged instruction that may be used by an operating system to allocate an unmeasured, zeroed page in EPC 220 to an existing secure enclave. An EMOD instruction is a privileged instruction that may be used by an operating system to modify the access rights or type associated with a previously allocated page in EPC 220. An EACCEPT instruction is an unprivileged instruction that may be used by an application inside a secure enclave to confirm changes made to the access rights or type of a page in EPC 220. An EREMOVE instruction is a privileged instruction that may be used by an operating system to de-allocate a page in EPC 220

from a secure enclave.

Secure enclave unit 200 may also include EPC logging unit 250. EPC logging unit 250 may include any microcode, logic, circuitry, or other hardware to provide for logging according to embodiments of the present invention. In one embodiment, the microcode or other control
5 logic for certain instructions causes logging to be performed when the instruction is executed by guest software running in a virtual machine (VM), instead of causing a VM exit. The instructions for which logging is performed include instructions that are related to the creation of, memory management in, and paging in secure enclaves. For example, the instructions may include ECREATE, EADD, EREMOVE, EAUG, EMOD, ELD, EWB, and EPA. Therefore,
10 embodiments of the present invention may provide for a virtual machine monitor (VMM) to monitor the memory management and paging activities of its guest software with improved performance gained by reducing the number of transfers of control between the guest and the VMM.

In one embodiment, logging is performed, for each instruction that references EPC 220, by
15 writing certain information to a page (e.g., log page 122) in system memory 120. The address of log page 122 is specified in an entry (e.g., entry 125) in the VM control structure (e.g., VMCS 124) of the VM in which the guest is running, and log page 122 is accessible to the VMM. Therefore, the VMM may have a record of EPC references so it may later perform updates to its internal data structures based on the logged information rather than requiring a VM exit for each
20 reference.

The information logged for each instruction may include the instruction (or leaf) and the EPC address associated with the instruction. Further information may be logged depending on the instruction. For example, for ECREATE, EADD, EAUG, EMOD, ELD, and EPA, the page type may be logged. For ECREATE, the state of the debug flag in the secure enclave control
25 structure (SECS) may be logged. For EADD, EAUG, and ELD, the SECS host physical address (HPA) may be logged.

The information may be logged in any desired format on log page 122. For example, each log entry may be 128 bits, where the instruction, the EPC address, and the page type are stored in an instruction field, an EPC address field, and a page type field, respectively, in bits 63:0. Other
30 instruction-dependent information (e.g., page type, debug flag, and SECS HPA) may be stored in one or more instruction-dependent fields in bits 127:64.

Figure 3 illustrates method 300 for logging in secure enclaves according to an embodiment of the present invention. Although method embodiments of the invention are not limited in this respect, reference may be made to elements of Figures 1 and 2 to help describe the method

embodiment of Figure 3. Note that all of method 300 between and including blocks 320 and 370 may occur in a virtual machine without a VM exit.

In box 310, a VM entry is performed to transfer control of processor 110 from a VMM running outside a VM to a guest operating system running inside the VM.

5 In box 320, creation of a secure enclave in the virtual machine may begin, for example, by the guest operating system using an ECREATE instruction. In box 322, the ECREATE instruction, the associated EPC address, the page type, and the debug flag are logged. In box 324, pages in EPC 220 may be allocated to the secure enclave, for example, by the guest operating system using an EADD instruction. In box 326, the EADD instruction, the associated
10 EPC address, the page type, and the SECS HPA are logged. In box 328, the secure enclave may be initiated, for example by the guest operating system using an EINIT instruction.

In box 330, a guest application may enter the secure enclave, for example by using an EENTER instruction. In box 332, the guest application in the secure enclave may request additional memory space in EPC 220. In box 334, the guest operating system may use and
15 EAUG instruction to allocate a first page in EPC 220 to the guest application. In box 336, the EAUG instruction, the associated EPC address, the page type, and the SECS HPA are logged. In box 338, the guest application in the secure enclave may confirm the allocation using an EACCEPT instruction.

In box 340, the guest operating system may use an EPA instruction to allocate a second
20 page in EPC 220 as a version array page. In box 342, the EPA instruction, the associated EPC address, and the page type are logged. In box 344, the guest operating system may use an EWB instruction to page a third page out of EPC 220. In box 346, the EWB instruction and the associated EPC address are logged.

In box 350, the guest operating system may use an ELD instruction to page a fourth page
25 into EPC 220. In box 352, the ELD instruction, the associated page type, and the SECS HPA are logged.

In box 360, the guest application in the secure enclave may request the guest operating system to free a fifth page in EPC 220. In box 362, the guest operating system may use an EMOD instruction to de-allocate the fifth page from the guest application. In box 364, the
30 EMOD instruction, the associated EPC address, and the page type are logged. In box 366, the guest application may use an EACCEPT instruction to confirm the allocation.

In box 370, the guest application may continue to execute in the secure enclave.

In box 380, control may be transferred to the VMM as the result of a VM exit. In box 382, the VMM may use the logged information to update its internal data structures.

In various embodiments of the present invention, the method illustrated in Figure 3 may be performed in a different order, with illustrated boxes combined or omitted, with additional boxes added, or with a combination of reordered, combined, omitted, or additional boxes.

Furthermore, many other method embodiments are possible within the scope of the present invention.

Embodiments or portions of embodiments of the present invention, as described above, may be stored on any form of a machine-readable medium. For example, all or part of method 300 may be embodied in software or firmware instructions that are stored on a medium readable by processor 110, which when executed by processor 110, cause processor 110 to execute an embodiment of the present invention. Also, aspects of the present invention may be embodied in data stored on a machine-readable medium, where the data represents a design or other information usable to fabricate all or part of processor 110.

Thus, embodiments of an invention for logging in secure enclaves have been described. While certain embodiments have been described, and shown in the accompanying drawings, it is to be understood that such embodiments are merely illustrative and not restrictive of the broad invention, and that this invention not be limited to the specific constructions and arrangements shown and described, since various other modifications may occur to those ordinarily skilled in the art upon studying this disclosure. In an area of technology such as this, where growth is fast and further advancements are not easily foreseen, the disclosed embodiments may be readily modifiable in arrangement and detail as facilitated by enabling technological advancements without departing from the principles of the present disclosure or the scope of the accompanying claims.

What is claimed is:

1. A processor comprising:

an instruction unit to receive an instruction having an associated enclave page cache
address; and
an execution unit to execute the instruction without causing a virtual machine exit,
wherein execution of the instruction includes logging the instruction and the
associated enclave page cache address.

2. The processor of claim 1, wherein execution of the instruction also includes creating a
secure enclave.

3. The processor of claim 1, wherein execution of the instruction also includes allocating a
page in the enclave page cache to a secure enclave.

4. The processor of claim 1, wherein execution of the instruction also includes allocating a
page in the enclave page cache as a version array page.

5. The processor of claim 1, wherein execution of the instruction also includes evicting a
page from the enclave page cache.

6. The processor of claim 1, wherein execution of the instruction also includes loading a
page into the enclave page cache.

7. The processor of claim 1, wherein execution of the instruction also includes de-
allocating a page from a secure enclave.

8. The processor of claim 2, wherein execution of the instruction also includes logging an
associated page type and a debug flag.

9. The processor of claim 3, wherein execution of the instruction also includes logging an
associated page type and an address of a control structure for the secure enclave.

10. The processor of claim 4, wherein execution of the instruction also includes logging

an associated page type.

11. The processor of claim 6, wherein execution of the instruction also includes logging an associated page type and an address of a control structure for a secure enclave.

5

12. The processor of claim 7, wherein execution of the instruction also includes logging an associated page type.

13. A method comprising:

10

receiving an instruction from a guest operating within a virtual machine, the instruction having an associated enclave page cache address; and

executing the instruction without causing an exit from the virtual machine, wherein execution of the instruction includes logging the instruction and the associated enclave page cache address.

15

14. The method of claim 13, wherein logging includes writing to a log page in memory, wherein the address of the log page is specified by an entry in a control structure for the virtual machine.

20

15. The method of claim 14, further comprising accessing, by a virtual machine monitor, the log page.

16. The method of claim 13, wherein logging also includes logging an associated page type.

25

17. The method of claim 13, wherein logging also includes logging a debug flag.

18. The method of claim 13, wherein logging also includes logging an address of a control structure for a secure enclave.

30

19. A system comprising:

a memory; and

a processor including

an instruction unit to receive an instruction having an associated enclave page cache

address; and

an execution unit to execute the instruction without causing a virtual machine exit, wherein execution of the instruction includes logging the instruction and the associated enclave page cache address in the memory.

5

20. The system of claim 19, wherein logging includes writing to a log page in the memory, wherein the address of the log page is specified by an entry in a control structure for the virtual machine.

10

FIGURE 1

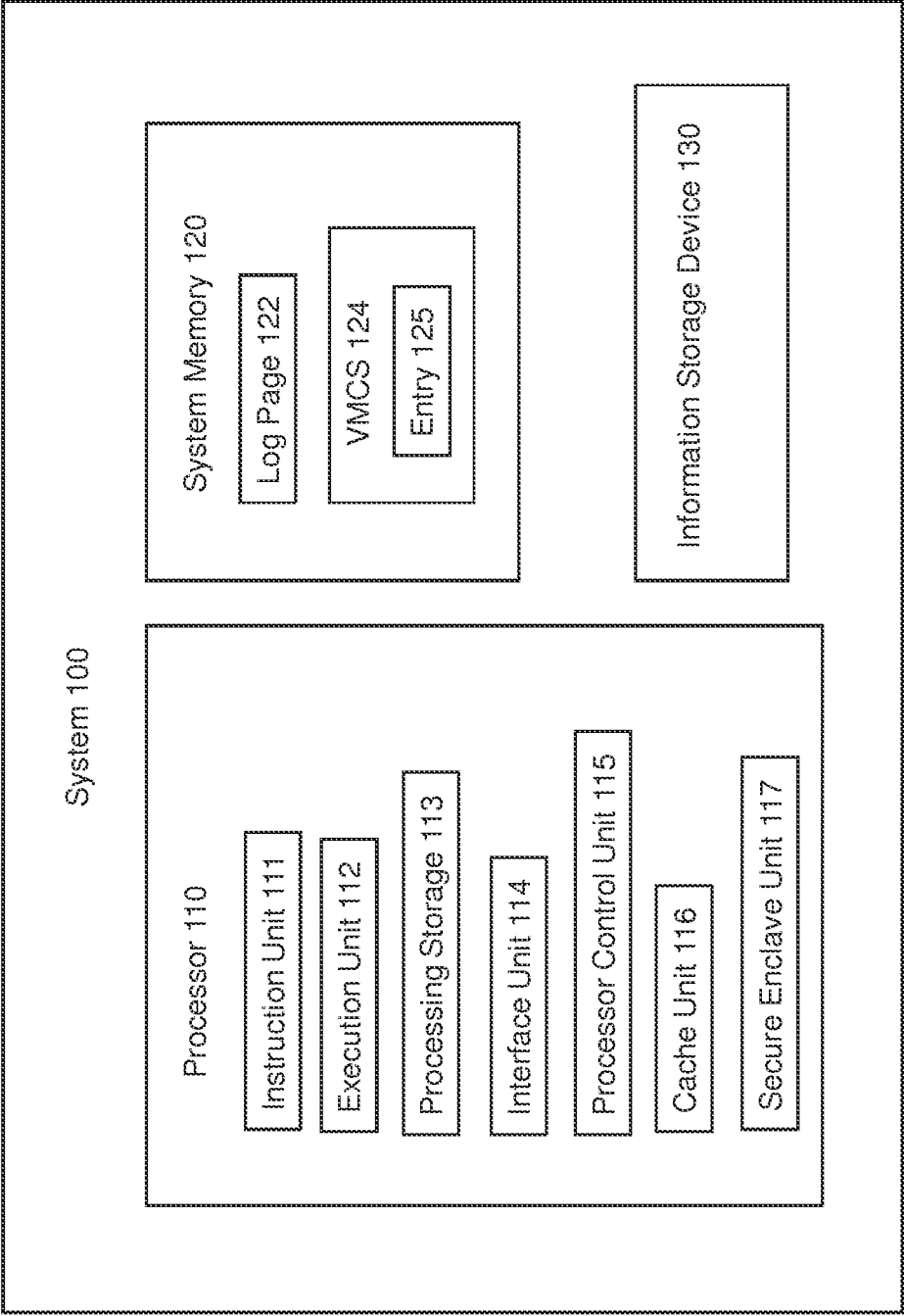


FIGURE 2

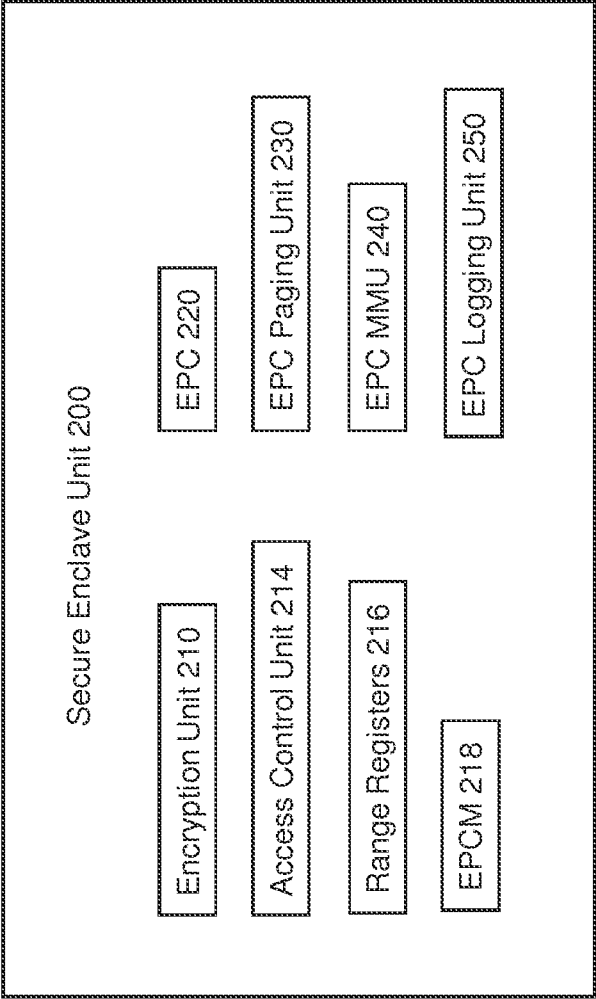
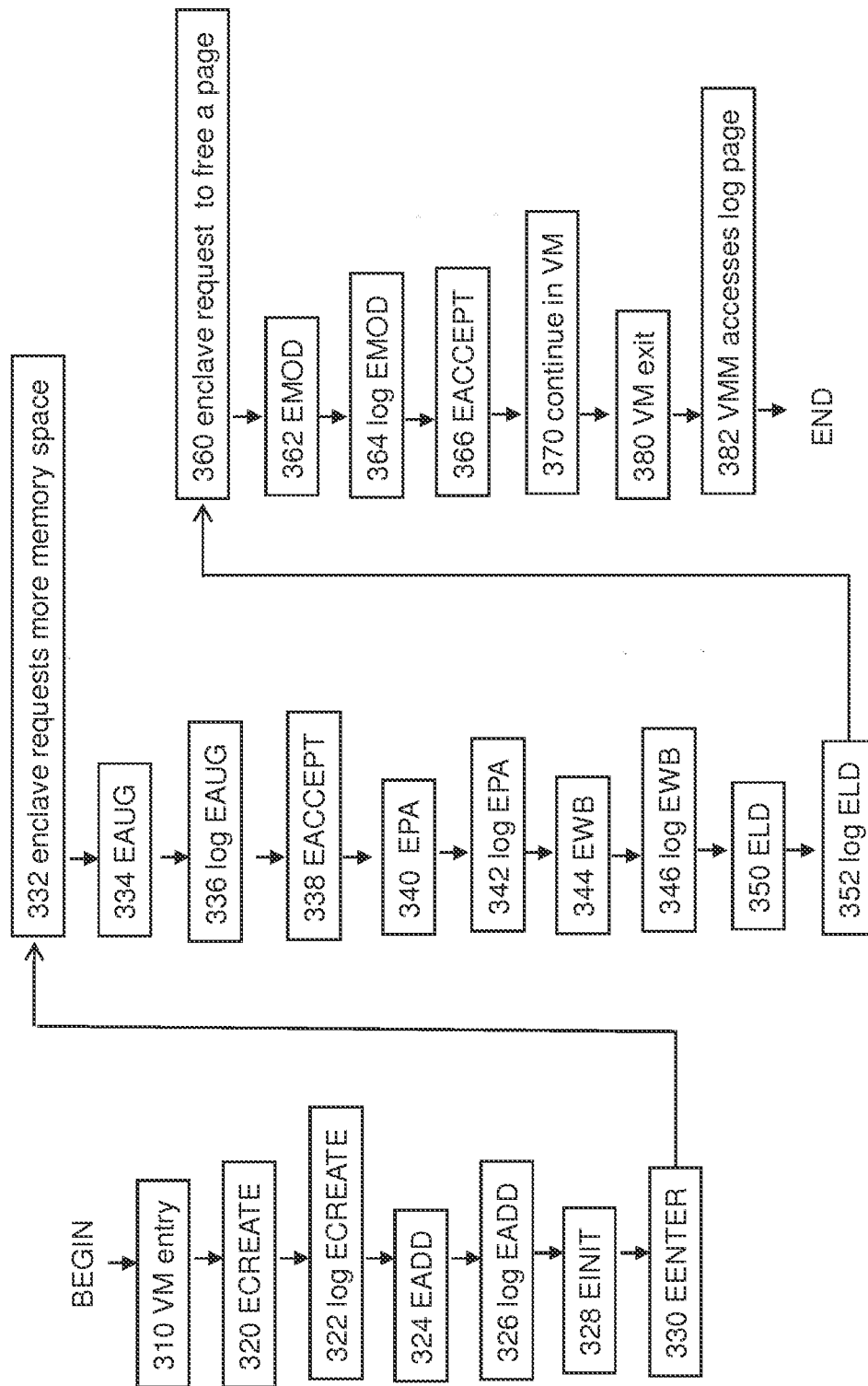


FIGURE 3
METHOD 300



A. CLASSIFICATION OF SUBJECT MATTER**G06F 12/14(2006.01)i, G06F 12/08(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 12/14; G06F 21/24; G06F 11/34; G06F 9/06; H04L 9/30; H04L 9/32; H04L 9/00; G06F 21/00; G06F 11/36; G06F 12/08

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: enclave page cache, log, instruction, processor, execute, allocate, deallocate.

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
Y	US 2012-0159184 A1 (SIMON P. JOHNSON et al.) 21 June 2012 See paragraphs [0037]-[0046], [0059]-[0067], [0078]-[0081], [0132]-[0139], [0154], [0161], [0163], [0185], [0331]-[0332], [0359]-[0363], [0367], [0369], [0374]-[0376], and [0599]; and figures 1-3.	1-13, 16-19
A		14-15, 20
Y	US 2010-0153784 A1 (HIROAKI INOUE et al.) 17 June 2010 See paragraphs [0037]-[0051] and claims 1-6.	1-13, 16-19
A	WO 2010-057065 A2 (INTEL CORPORATION) 20 May 2010 See page 1, line 29 -page 3, line 5; claims 1-8; and figures 1-3.	1-20
A	US 2012-0163589 A1 (SIMON P. JOHNSON et al.) 28 June 2012 See paragraphs [0037]-[0049] and [0075]-[0085]; claims 1-9; and figures 1-3.	1-20
A	US 2005-0100163 A1 (MARK BUER) 12 May 2005 See paragraphs [0010]-[0037]; claims 1-7; and figures 1-10.	1-20
A	US 2007-0277223 A1 (SHAMANNA M. DATTA et al.) 29 November 2007 See paragraphs [0016]-[0017], [0030], and [0035]; and figure 3.	1-20



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 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

25 October 2013 (25.10.2013)

Date of mailing of the international search report

28 October 2013 (28.10.2013)

Name and mailing address of the ISA/KR

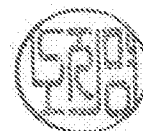

 Korean Intellectual Property Office
 189 Cheongsa-ro, Seo-gu, Daejeon Metropolitan City,
 302-701, Republic of Korea

Facsimile No. +82-42-472-7140

Authorized officer

NHO, Ji Myong

Telephone No. +82-42-481-8528



INTERNATIONAL SEARCH REPORT

Information on patent family members

International application No.

PCT/US2013/047332

Patent document cited in search report	Publication date	Patent family member(s)	Publication date
US 2012-0159184 A1	21/06/2012	US 2013-232344 A1 WO 2012-082410 A2 WO 2012-082410 A3	05/09/2013 21/06/2012 16/08/2012
US 2010-0153784 A1	17/06/2010	US 8140912 B2 WO 2008-053723 A1	20/03/2012 08/05/2008
WO 2010-057065 A2	20/05/2010	JP 2012-508938 A WO 2010-057065 A3	12/04/2012 19/08/2010
US 2012-0163589 A1	28/06/2012	US 2013-232345 A1 WO 2012-087562 A2 WO 2012-087562 A3	05/09/2013 28/06/2012 16/08/2012
US 2005-0100163 A1	12/05/2005	CN 100542085 C CN 1677922 A EP 1536308 A2 EP 1536308 A3 TW I298591B US 2010-0241841 A1 US 7734932 B2	16/09/2009 05/10/2005 01/06/2005 31/05/2006 01/07/2008 23/09/2010 08/06/2010
US 2007-0277223 A1	29/11/2007	CN 101454751 A DE 112007001321 T5 JP 2009-534763 A JP 2012-089147 A JP 4883459 B2 KR 10-1263061 B1 KR 10-2009-0005219 A TW 200820081 A US 2013-212672 A1 WO 2007-140300 A1	10/06/2009 23/04/2009 24/09/2009 10/05/2012 22/02/2012 09/05/2013 12/01/2009 01/05/2008 15/08/2013 06/12/2007