



- (51) **International Patent Classification:**
G06F 9/312 (2006.01)
- (21) **International Application Number:**
PCT/US2017/014262
- (22) **International Filing Date:**
20 January 2017 (20.01.2017)
- (25) **Filing Language:** English
- (26) **Publication Language:** English
- (30) **Priority Data:**
62/286,280 22 January 2016 (22.01.2016) US
- (71) **Applicant:** SONY INTERACTIVE ENTERTAINMENT INC [JP/JP]; 1-7-1 Konan, Minato-Ku, Tokyo 108-0075 (JP).
- (72) **Inventors; and**
- (71) **Applicants :** CERNY, Mark Evan [US/US]; c/o Sony Interactive Entertainment America LLC, 2207 Bridgepointe Parkway, San Mateo, California 94404 (US). PILGRIM, Simon [GB/GB]; c/o Sony Interactive Entertainment Europe Limited, 10 Marlborough Street, London Middlesex W1F 7LP (GB).
- (74) **Agent:** ISENBERG, Joshua D.; c/o JDI Patent, 809 Corporate Way, Fremont, California 94539 (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, KH, 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:** SPOOFING CPUID FOR BACKWARDS COMPATIBILITY

100

(57) **Abstract:** To facilitate backwards compatibility, a computing device may respond to a call from an application for information regarding a processor on the computing device by returning information regarding a different processor than the processor on the computing device.

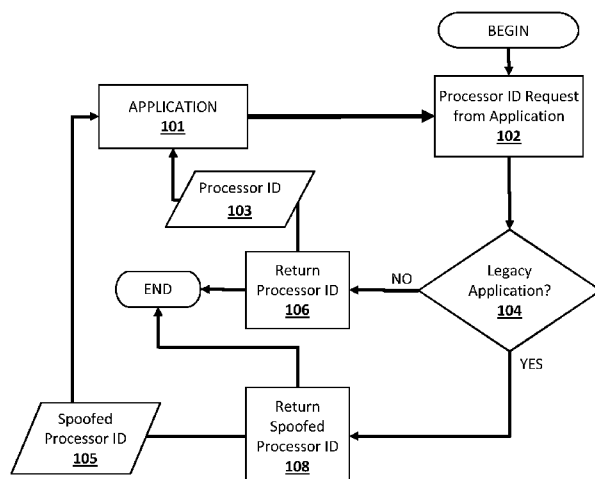


FIG. 1



Published:

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

SPOOFING CPUID FOR BACKWARDS COMPATIBILITY

PRIORITY CLAIM

This application claims the benefit of prior to commonly-assigned, US Provisional application number 62/286,280, filed January 22, 2016 the entire contents of which are herein incorporated by reference.

BACKGROUND

When a new version of a computer system (a “new device”) is released it is desirable for the applications written for a previous version of the system (a “legacy device”) to run flawlessly on the new device. This capability is often referred to as “backwards compatibility” with respect to “legacy applications”. Even if the new device is capable of executing legacy applications created for the legacy device, the new device may still fail to be backwards compatible when running those applications. Differences in performance of the hardware components of the new device and legacy device can cause errors in synchronization on the new device. Such differences in performance can arise, e.g., from differences in capabilities of the central processing unit (CPU) of the new device relative to the legacy device. For example, if the CPU on the new device is faster than the CPU on the legacy device, data still being used by another component of the new device may be prematurely overwritten by the CPU.

BRIEF DESCRIPTION OF THE DRAWINGS

The teachings of the present disclosure can be readily understood by considering the following detailed description in conjunction with the accompanying drawings, in which:

FIG. 1 is a flow diagram illustrating a method for implementing backwards compatibility according to aspects of the present disclosure.

FIG. 2 is a block diagram depicting a central processing unit (CPU) in which executable software for carrying out the method of FIG. 1 is implemented in Read-Only Memory (ROM).

FIG. 3 is a block diagram of a device that includes a CPU of the type shown in FIG. 2.

SUMMARY OF THE INVENTION

The disadvantages associated with the prior art are overcome by aspects of the present disclosure relating to a method in which a computing device responds to a call from an application for information regarding a processor on the computing device by returning information regarding a different processor than the processor on the computing device.

DETAILED DESCRIPTION

Although the following detailed description contains many specific details for the purposes of illustration, anyone of ordinary skill in the art will appreciate that many variations and alterations to the following details are within the scope of the claimed invention. Accordingly, the exemplary embodiments of the invention described below are set forth without any loss of generality to, and without imposing limitations upon, the claimed invention.

Introduction

To address problems resulting from differences in CPU behavior relative to a legacy device, a new device may mimic certain capabilities of the legacy device CPU when running legacy applications. A key feature of the ability to mimic a legacy device is to fool the legacy application into acting as though it is running on a legacy device. Since many applications are designed to be run on different processors, modern processors often implement an opcode or a register that allows a software application to discover details of the processor. To facilitate backwards compatibility, a processor on a new device can execute the opcode or supply the register value in such a way that different processor information is returned in response to a call from a legacy application. The different information is referred to herein as a “Spoofed Processor ID”. The spoofed processor ID would selectively identify certain features of the new device as being either different from ones that are actually supported or not being supported at all when in fact they are.

Method

The method **100** depicted in the flow diagram shown in FIG. 1 illustrates an example of how this would work. Specifically, a requesting application **101** running on a device requests a processor ID from the processor running the application, as indicated at **102**. The processor determines at

104 whether the requesting application **101** is a legacy application, i.e., one originally written for an earlier version of the device. There are a number of ways in which the processor could determine whether the application is written for the new device or for a legacy device. One way would be if all applications written for the new device reported information to the processor upon loading that identified the application as being written for the new device. The reporting of such information would identify the application as having been written for the new device and failure to report such information would identify the application as a legacy application.

The information returned by the processor to the requesting application **101** would vary based on whether the application **101** is a legacy application or a new application, i.e., one written to run on the device. If the requesting application **101** is a new application, the processor returns true processor information, e.g., the correct processor ID **103** for the processor on the device running the application, as indicated at **106**. If the requesting application **101** is a legacy application, the processor returns spoofed processor information, e.g., a spoofed processor ID **105**, as indicated at **108**. The returned information may identify certain features that are in fact supported by the processor as not being supported, or specify that the processor running the application is a legacy processor.

By way of example and not by way of limitation, the spoofed processor ID **105** may be returned by a modified CUID instruction, which is an opcode supported by the x86 architecture. By using the CUID opcode, software can determine processor type and the presence of features such as implementation of the various SSE instruction set architectures. On the x86 architecture the CUID opcode is the bytes 0Fh and A2h and the value in the EAX register, and in some cases the ECX register, specifies what information to return.

In assembly language the CUID instruction takes no parameters and instead implicitly uses the EAX register to determine the main category of information returned, which is commonly called the CUID leaf. An application calling CUID with EAX=0 will receive in the EAX register the highest EAX calling parameter (leaf) that the CPU supports in return, and other registers contain a vendor ID that identifies the CPU manufacturer. To obtain extended function information CUID may be called with the most significant bit of EAX set. To determine the highest

extended function calling parameter, CPUID may be called with EAX = 80000000h. Some leaves also have sub-leaves, which are selected via the ECX register before calling CPUID.

Calling CPUID with EAX=1 returns information on the CPU's stepping model, model, and family information in EAX (also called the *signature* of a CPU), feature flags in EDX and ECX, and additional feature info in EBX. Calling CPUID with EAX=2 returns a list of descriptors indicating cache and translation lookaside buffer (TLB) capabilities in EAX, EBX, ECX and EDX registers. Other EAX values return information such as processor serial number, thread/core and cache topology, extended features, processor brand, L1 cache and TLB identifiers, extended L2 cache features, power management feature identifiers, and virtual and physical address sizes. A program of the type shown in FIG. 1 may return values that are different for the correct values of any or all of these if queried by a legacy application.

CPU

The processor ID spoofing capability shown in FIG. 1 can be implemented on a new device in any of a number of different ways. For example, a new device may include special microcode in ROM or RAM that returns the spoofed processor ID information in response to a query from a legacy application. FIG. 2 illustrates one possible implementation of a CPU **200** in which the microcode is implemented in a ROM that is part of a CPU core **202**. The CPU may include one or more such cores. The CPU cores **202** may share a common level 2 cache **204** and bus interface **206**. Each core **202** may contain logic units **208** for loading, storing, scheduling, decoding, and executing instructions along with registers **210**, caches **212**, and translation lookaside buffers (TLBs) **214** for instructions and data.

In the particular implementation shown in FIG. 2, the CPUID instruction is implemented by microcode **216** stored in a ROM **218** that contains a processor ID program and two different sets of processor ID data. One set of data (ID1) is for new device applications and the other set (ID2) is for legacy device applications.

In alternative implementations there may be two different processor ID programs, one for legacy applications and one for new device application. In such implementations, the CPU **200** may selectively run one program or the other when executing the CUID instruction depending on whether the application it is running is a legacy application or a new device application. In other alternative implementations, the processor ID program may be implemented by special dedicated hardware, e.g., certain aspects of returning the processor ID information may be implemented by hardware logic rather than microcode stored in ROM **218**. In other alternative implementations, the microcode may be stored in Random Access Memory (RAM) rather than in ROM.

System

A CPU of the type shown in FIG. 2 may be incorporated into a new device. As seen in FIG. 3, the new device **300** may include a CPU **200** with multiple cores **202** and a GPU **302** with multiple compute units (CUs) **304**. The CPU and GPU may communicate with each other and with a memory and I/O controller **310** through L2 caches **204**, **304**, bus interfaces **206**, **306** and a frontside bus **308**. The memory and I/O controller **310** manages data transfers to and from the CPU **200** and GPU **302** and a memory **312** (e.g., RAM), storage devices **314** (e.g., disk drives or flash memory), video cards **316**, peripherals **318**, and a network **320** via appropriately configured busses **322**.

The device **300** runs the application **101** by executing its instructions on the CPU **200** and GPU **302**. Portions of the application **101** may be loaded into memory **312**. In one particular implementation, the CUID instruction is implemented by microcode that contains a processor ID program and two different sets of processor ID data, e.g., as discussed above with respect to FIG. 2.

As noted above, in alternative implementations there may be two different processor ID programs, one for legacy applications and one for new device application. In such implementations, the CPU **200** may selectively run one program or the other when executing the CUID instruction depending on whether the application it is running is a legacy application or a new device application. In other alternative implementations, a single processor ID program

may return processor ID data that differs in some specifics depending on whether the querying program is a legacy application or a new device application. In other alternative implementations, the processor ID code **216** may be implemented by special dedicated hardware, e.g., certain aspects of returning the processor ID information may be implemented by hardware logic rather than microcode stored in a ROM **218** of a CPU core **202**. In other alternative implementations, the microcode may be stored in Random Access Memory (RAM), e.g., main memory **312** rather than in ROM.

Returning spoofed processor feature information can facilitate resolution of backward compatibility issues by preventing an application from taking advantage of features that, even though they are supported, might cause timing issues if a legacy application attempts to use them.

While the above is a complete description of the preferred embodiments of the present invention, it is possible to use various alternatives, modifications, and equivalents. Therefore, the scope of the present invention should be determined not with reference to the above description but should, instead, be determined with reference to the appended claims, along with their full scope of equivalents. Any feature, whether preferred or not, may be combined with any other feature, whether preferred or not. In the claims that follow, the indefinite article “A” or “An” refers to a quantity of one or more of the item following the article, except where expressly stated otherwise. The appended claims are not to be interpreted as including means-plus-function limitations, unless such a limitation is explicitly recited in a given claim using the phrase “means for”. Any element in a claim that does not explicitly state “means for” performing a specified function, is not to be interpreted as a “means” or “step” clause as specified in 35 USC § 112(f).

CLAIMS

- 1 1. A method, comprising:
2 in a computing device, responding to a call from an application for information regarding a
3 processor on the computing device by returning information regarding a different processor
4 than the processor on the computing device.
- 1 2. The method of claim 1, wherein the information regarding the different processor identifies
2 certain features of the processor on the computing device as being different from ones that
3 are actually supported by the computing device.
- 1 3. The method of claim 1, wherein the information regarding the different processor identifies
2 certain features of the processor on the computing device as not being supported at all by the
3 processor on the computing device when in fact the certain features are supported by the
4 processor on the computing device.
- 1 4. The method of claim 1, wherein the application is a legacy application written for a legacy
2 computing device.
- 1 5. The method of claim 1, wherein the application is a legacy application written for a legacy
2 computing device and wherein the information regarding the different processor identifies
3 certain features of the processor on the computing device as being different from ones that
4 are actually supported by the computing device.
- 1 6. The method of claim 1, wherein the application is a legacy application written for a legacy
2 computing device and wherein the information regarding the different processor identifies
3 certain features of the processor on the computing device as not being supported at all by the
4 processor on the computing device when in fact the certain features are supported by the
5 processor on the computing device.
- 1 7. The method of claim 1, wherein the application is a legacy application written for a legacy
2 computing device and wherein the information regarding the different processor identifies a
3 processor on the legacy device.
- 1 8. The method of claim 1, wherein returning the information regarding the different processor
2 includes use of an opcode supported by an architecture of the processor on the computing
3 device.

- 1 9. The method of claim 1, wherein returning the information regarding the different processor
2 includes use of an opcode supported by an x86 architecture.
- 1 10. The method of claim 1, wherein returning the information regarding the different processor
2 includes use of a CPUID instruction.
- 1 11. The method of claim 1, wherein returning the information regarding the different processor
2 includes use of a microcode stored in a ROM or RAM associated with the processor on the
3 computing device.
- 1 12. The method of claim 1, wherein returning the information regarding the different processor
2 includes use of a microcode stored in a ROM or RAM associated with the processor on the
3 computing device and two different sets of processor ID data, wherein one set of data is for
4 new device applications and another other set of data is for legacy device applications.
- 5 13. The method of claim 1, wherein returning the information regarding the different processor
6 includes use of a microcode stored in a ROM or RAM associated with the processor on the
7 computing device, and the microcode logic returns processor ID data that differs in some
8 specifics depending on whether the querying program is a new device application or a legacy
9 device application
- 10 14. The method of claim 1, wherein returning the information regarding the different processor
11 includes use of one ID program for legacy applications and another ID program for new
12 device applications.
- 1 15. The method of claim 1, wherein returning the information regarding the different processor
2 includes use of dedicated hardware.
- 1 16. A computing device, comprising:
2 a processor configured to respond to a call from an application for information regarding the
3 processor by returning information regarding a different processor.
- 1 17. The computing device of claim 16, wherein the information regarding the different processor
2 identifies certain features of the processor on the computing device as being different from
3 ones that are actually supported by the computing device.
- 1 18. The computing device of claim 16, wherein the information regarding the different processor
2 identifies certain features of the processor on the computing device as not being supported at

all by the processor on the computing device when in fact the certain features are supported by the processor on the computing device.

19. The computing device of claim 16, wherein the application is a legacy application written for a legacy computing device.

20. The computing device of claim 16, wherein the application is a legacy application written for a legacy computing device and wherein the information regarding the different processor identifies certain features of the processor on the computing device as being different from ones that are actually supported by the computing device.

21. The computing device of claim 16, wherein the application is a legacy application written for a legacy computing device and wherein the information regarding the different processor identifies certain features of the processor on the computing device as not being supported at all by the processor on the computing device when in fact the certain features are supported by the processor on the computing device.

22. The computing device of claim 16, wherein the application is a legacy application written for a legacy computing device and wherein the information regarding the different processor identifies a processor on the legacy device.

23. The computing device of claim 16, wherein returning the information regarding the different processor includes use of an opcode supported by an architecture of the processor on the computing device.

24. The computing device of claim 16, wherein returning the information regarding the different processor includes use of an opcode supported by an x86 architecture.

25. The computing device of claim 16, wherein returning the information regarding the different processor includes use of a CPUID instruction.

26. The computing device of claim 16, further comprising a ROM or RAM associated with the processor on the computing device, wherein returning the information regarding the different processor includes use of a microcode stored in the ROM or RAM associated with the processor on the computing device.

27. The computing device of claim 16, further comprising a ROM or RAM associated with the processor on the computing device, wherein returning the information regarding the different

processor includes use of a microcode stored in the ROM or RAM associated with the processor on the computing device and two different sets of processor ID data, wherein one set of data is for new device applications and another other set of data is for legacy device applications.

28. The computing device of claim 16, further comprising a ROM or RAM associated with the processor on the computing device, wherein returning the information regarding the different processor includes use of a microcode stored in a ROM or RAM associated with the processor on the computing device, and the microcode logic returns processor ID data that differs in some specifics depending on whether the querying program is a new device application or a legacy device application.

29. The computing device of claim 16, wherein returning the information regarding the different processor includes use of one ID program for legacy applications and another ID program for new device applications.

30. The computing device of claim 16, further comprising dedicated hardware configured to return the information regarding the different processor includes use of dedicated hardware.

31. A non-transitory computer-readable medium having embodied therein executable instructions configured to cause a computing device to respond to a call from an application for information regarding a processor on a computing device by returning information regarding a different processor than the processor on the computing device when executed.

32. The non-transitory computer-readable medium of claim 31, wherein the information regarding the different processor identifies certain features of the processor on the computing device as being different from ones that are actually supported by the computing device.

33. The non-transitory computer-readable medium of claim 31, wherein the information regarding the different processor identifies certain features of the processor on the computing device as not being supported at all by the processor on the computing device when in fact the certain features are supported by the processor on the computing device.

34. The non-transitory computer-readable medium of claim 31, wherein the application is a legacy application written for a legacy computing device.

- 1 35. The non-transitory computer-readable medium of claim 31, wherein the application is a
2 legacy application written for a legacy computing device and wherein the information
3 regarding the different processor identifies certain features of the processor on the computing
4 device as being different from ones that are actually supported by the computing device.
- 1 36. The non-transitory computer-readable medium of claim 31, wherein the application is a
2 legacy application written for a legacy computing device and wherein the information
3 regarding the different processor identifies certain features of the processor on the computing
4 device as not being supported at all by the processor on the computing device when in fact
5 the certain features are supported by the processor on the computing device.
- 1 37. The non-transitory computer-readable medium of claim 31, wherein the application is a
2 legacy application written for a legacy computing device and wherein the information
3 regarding the different processor identifies a processor on the legacy device.
- 1 38. The non-transitory computer-readable medium of claim 31, wherein returning the
2 information regarding the different processor includes use of an opcode supported by an
3 architecture of the processor on the computing device.
- 1 39. The non-transitory computer-readable medium of claim 31, wherein returning the
2 information regarding the different processor includes use of an opcode supported by an x86
3 architecture.
- 1 40. The non-transitory computer-readable medium of claim 31, wherein returning the
2 information regarding the different processor includes use of a CPUID instruction.
- 1 41. The non-transitory computer-readable medium of claim 31, wherein returning the
2 information regarding the different processor includes use of a microcode stored in a ROM or
3 RAM associated with the processor on the computing device.
- 1 42. The non-transitory computer-readable medium of claim 31, wherein returning the
2 information regarding the different processor includes use of a microcode stored in a ROM or
3 RAM associated with the processor on the computing device and two different sets of
4 processor ID data, wherein one set of data is for new device applications and another other
5 set of data is for legacy device applications.

- 6 43. The non-transitory computer-readable medium of claim 31, wherein returning the
7 information regarding the different processor includes use of a microcode stored in a ROM or
8 RAM associated with the processor on the computing device, and the microcode logic returns
9 processor ID data that differs in some specifics depending on whether the querying program
10 is a new device application or a legacy device application
- 1 44. The non-transitory computer-readable medium of claim 31, wherein returning the
2 information regarding the different processor includes use of one ID program for legacy
3 applications and another ID program for new device applications.
45. The non-transitory computer-readable medium of claim 31, wherein returning the
information regarding the different processor includes use of dedicated hardware.

1/3

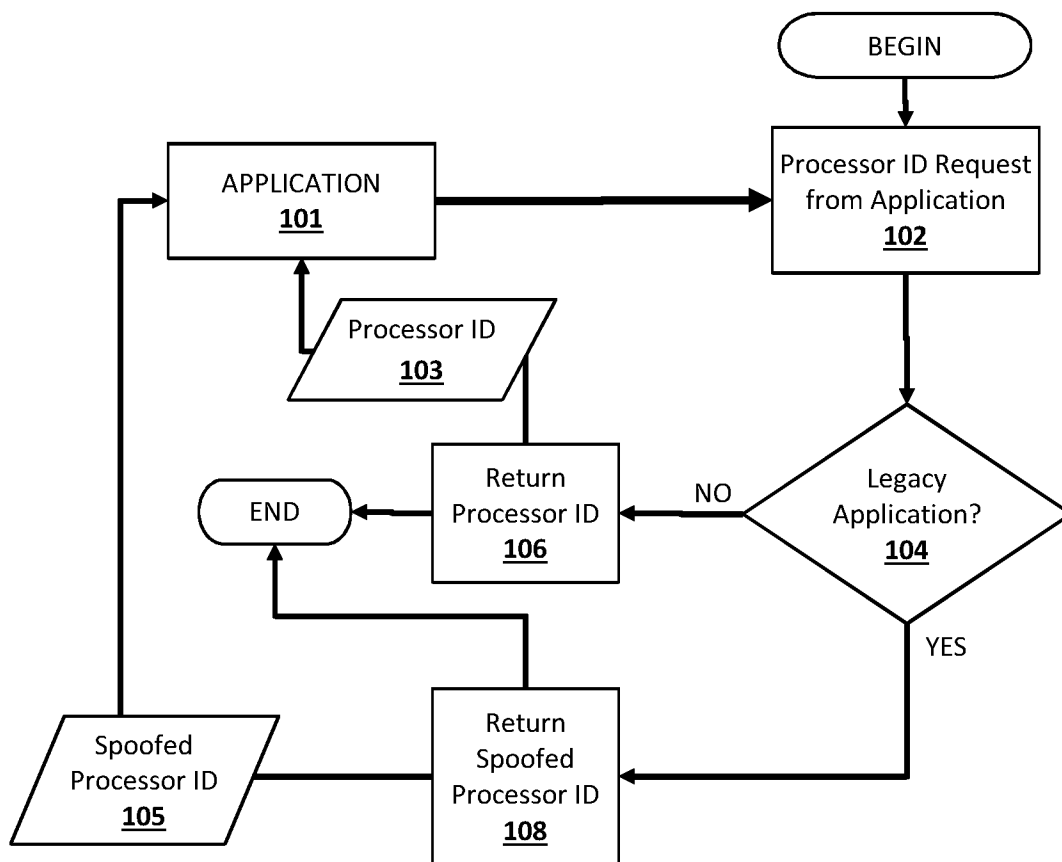
100

FIG. 1

2/3

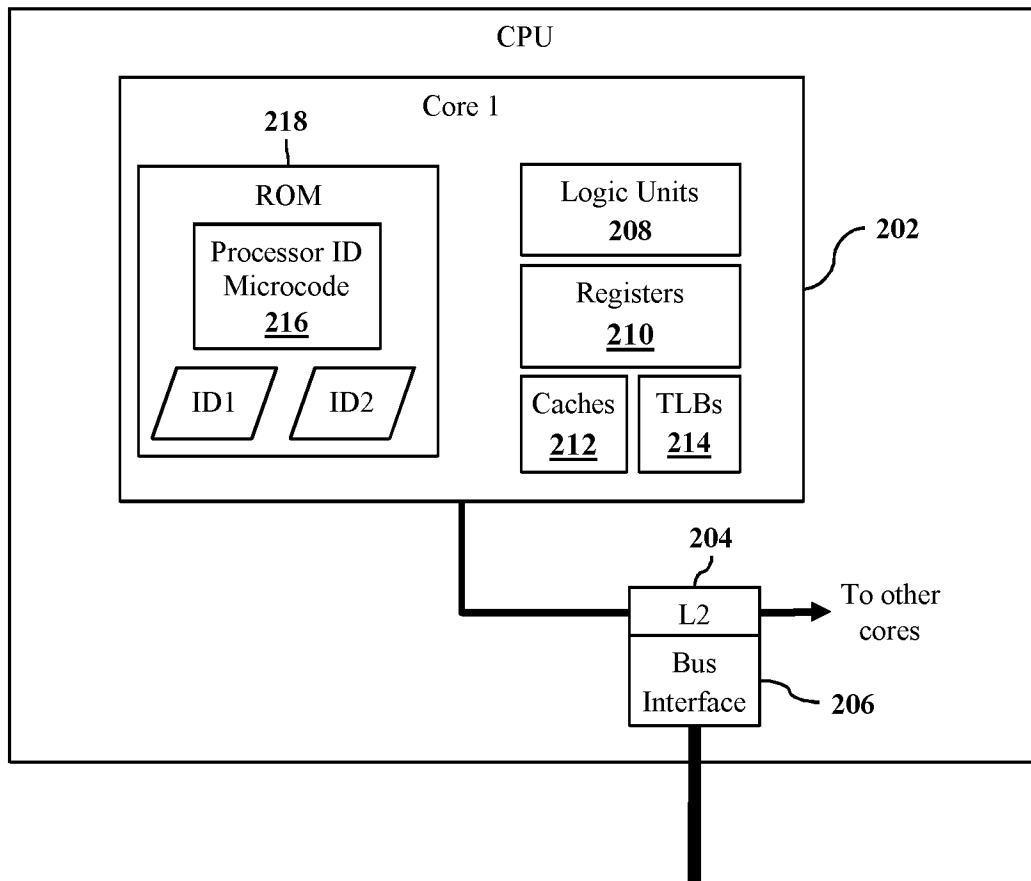
200

FIG. 2

3/3

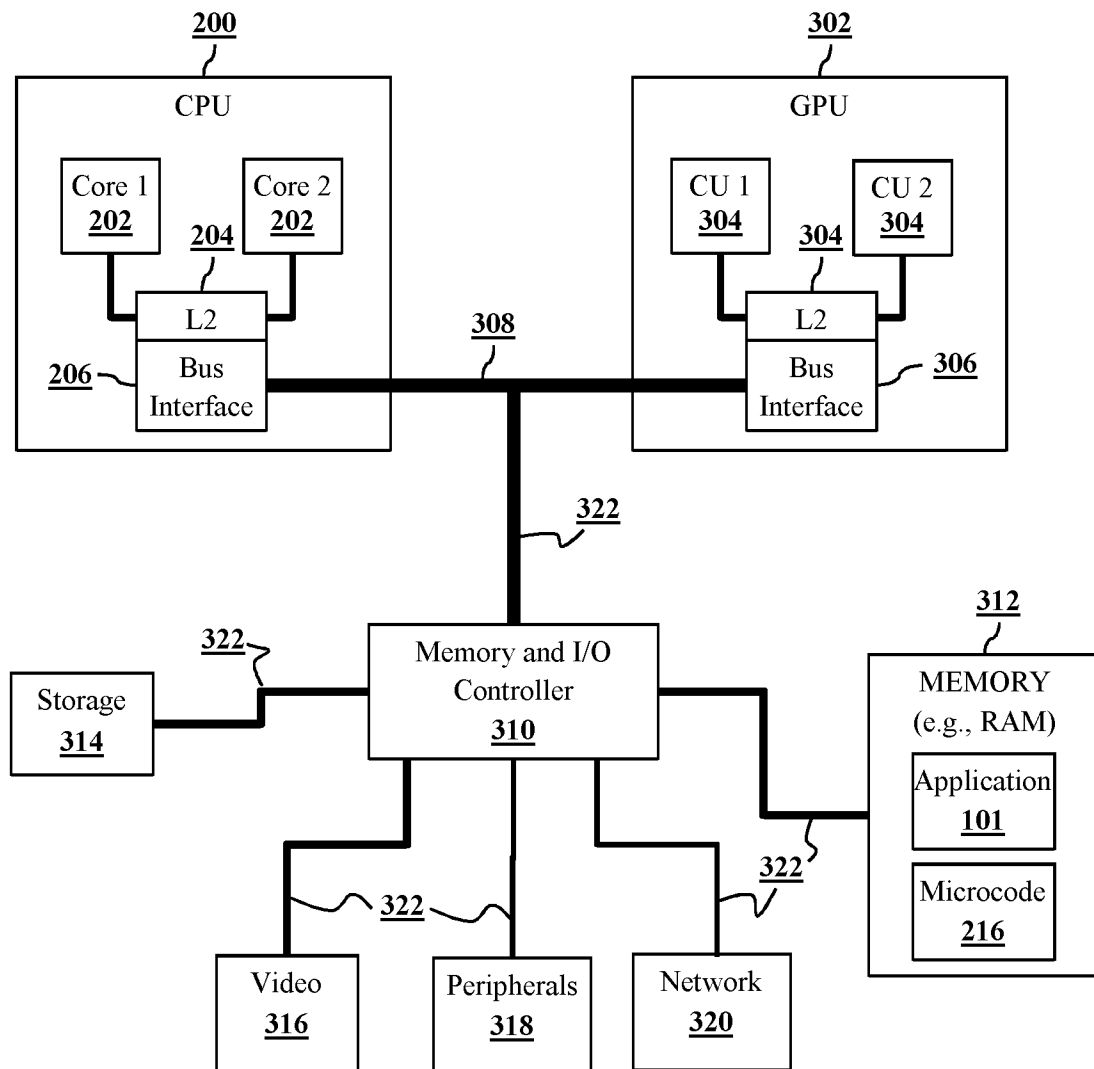
300

FIG. 3

INTERNATIONAL SEARCH REPORT

International application No.
PCT/US 17/14262

A. CLASSIFICATION OF SUBJECT MATTER
IPC(8) - G06F 9/312 (2017.01)
CPC - G06F 9/30036

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)

See Search History Document

Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched

See Search History Document

Electronic data base consulted during the international search (name of data base and, where practicable, search terms used)

See Search History Document

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
X -- Y	US 2009/0070760 A1 (KHATRI et al.), 12 March 2009 (12.03.2009), entire document, especially Abstract; Para [0019], [0021], [0029]-[0032], [0035]	1-3, 8-11, 13-18, 23-26, 28-33, 38-41 and 43-45 ----- 4-7, 12, 19-22, 27, 34-37 and 42
Y	US 2002/0112145 A1 (BIGBEE et al.), 15 August 2002 (15.08.2002), entire document, especially Abstract; Para [0003], [0035]-[0038], [0040]-[0042]	4-7, 12, 19-22, 27, 34-37 and 42
A	US 8,949,713 B1 (SUN et al.), 03 February 2015 (03.02.2015), entire document	1-45
A	US 2002/0055828 A1 (SWOBODA et al.), 09 May 2002 (09.05.2002), entire document	1-45
A	US 2009/0240980 A1 (INOUE et al.), 24 September 2009 (24.09.2009), entire document	1-45

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

* 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

20 March 2017

Date of mailing of the international search report

14 APR 2017

Name and mailing address of the ISA/US

Mail Stop PCT, Attn: ISA/US, Commissioner for Patents
P.O. Box 1450, Alexandria, Virginia 22313-1450
Facsimile No. 571-273-8300

Authorized officer:

Lee W. Young

PCT Helpdesk: 571-272-4300
PCT OSP: 571-272-7774