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

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
2 February 2017 (02.02.2017)

WIPO | PCT

(10) International Publication Number
WO 2017/019286 A1

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

(21) International Application Number:
PCT/US2016/041762

(22) International Filing Date:
11 July 2016 (11.07.2016)

(25) Filing Language: English

(26) Publication Language: English

(30) Priority Data:
14/810,361 27 July 2015 (27.07.2015) US

(71) Applicant: **SONY INTERACTIVE ENTERTAINMENT AMERICA LLC** [US/US]; 2207 Bridgepointe Parkway, San Mateo, California 94404 (US).

(72) Inventors: **CERNY, Mark Evan**; c/o Sony Interactive Entertainment America LLC, 2207 Bridgepointe Parkway, San Mateo, California 94404 (US). **SIMPSON, David**; c/o Sony Interactive Entertainment America LLC, 2207 Bridgepointe Parkway, San Mateo, California 94404 (US).

(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, 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, 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).

Published:

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

(54) Title: BACKWARD COMPATIBILITY BY RESTRICTION OF HARDWARE RESOURCES

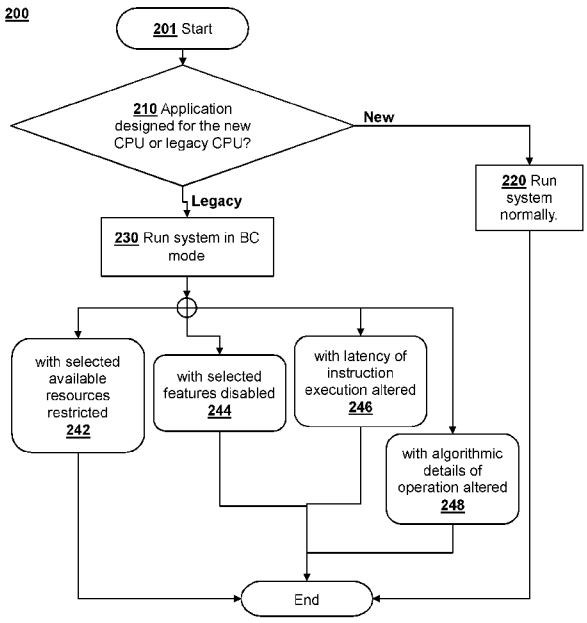

FIG. 2

(57) Abstract: A new device executing an application on a new central processing unit (CPU), determines whether the application is for a legacy device having a legacy CPU. When the new device determines that the application is for the legacy device, it executes the application on the new CPU with selected available resources of the new device restricted to approximate or match a processing behavior of the legacy CPU.

WO 2017/019286 A1

BACKWARD COMPATIBILITY BY RESTRICTION OF HARDWARE RESOURCES

This application claims the benefit of prior to commonly-assigned, US Provisional application number 14/810,361, filed July 27, 2015, the entire contents of which are herein incorporated by reference.

FIELD OF THE DISCLSOURE

Aspects of the present disclosure are related to execution of a computer application on a computer system. In particular, aspects of the present disclosure are related to a system or a method that provides backward compatibility for applications/titles designed for older versions of a computer system.

BACKGROUND

Modern computer systems often use a number of different processors for different computing tasks. For example, in addition to a number of central processing units (CPUs), a modern computer may have a graphics processing unit (GPU) dedicated to certain computational tasks in a graphics pipeline, or a unit dedicated to digital signal processing for audio, all of which are potentially part of an accelerated processing unit (APU) that may contain other units as well. These processors are connected to memory of various types, using buses that may be internal to an APU or externally located on the computer's motherboard.

It is common that a set of applications are created for a computer system such as a video game console or smartphone (the "legacy device"), and when a variant or a more advanced version of the computer system is released (the "new device") it is desirable for the applications of the legacy device to run flawlessly on the new device without recompilation or any modification that takes into account the properties of the new device. This aspect of the new device, as contained in its hardware architecture, firmware and operating system, is often referred to as "backwards compatibility."

Backwards compatibility is often achieved through binary compatibility, where the new device is capable of executing programs created for the legacy device. However, when the real time behavior of the category of devices is important to their operation, as is in the case of video game consoles or smartphones, significant differences in the speed of operation of a new device may cause it to fail to be backwards compatible with respect to a legacy device. If the new device is of lower performance than the legacy device, issues that prevent backwards

compatibility may arise; this is also true if the new device is of higher performance, or has different performance characteristics when compared to the legacy device.

It is within this context that aspects of the present disclosure arise.

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 block diagram illustrating an example of a central processing unit (CPU) core that may be configured to operate in a backwards compatibility mode in accordance with aspects of the present disclosure.

FIG. 2 is a flow diagram illustrating an example of a possible process flow for operating a CPU in a backwards compatibility mode in accordance with aspects of the present disclosure.

FIG. 3 is a block diagram of a device having a CPU configured to operate in a backwards compatibility mode in accordance with aspects of the present disclosure.

DESCRIPTION OF THE DRAWINGS

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 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

Even if the CPUs of the new device are binary compatible with the legacy device (i.e. capable of executing programs created for the legacy device), differences in performance characteristics between the CPUs of the new device and the CPUs of the legacy device may cause errors in legacy applications, and as a result the new device will not be backwards compatible.

If the CPUs of the new device have lower performance than the CPUs of the legacy device, many errors in a legacy application may arise due to the inability to meet real time deadlines imposed by display timing, audio streamout or the like. If the CPUs of the new device have substantially higher performance than the CPUs of the legacy device, many errors in a legacy

application may arise due to the untested consequences of such high speed operation. For example, in a producer-consumer model, if a consumer of data (e.g. the CPU) operates at higher speed than originally anticipated, it may attempt to access data before the data producer (e.g. some other component of the computer) makes it available. Alternatively if the producer of the data (e.g. the CPU) operates at higher speed than originally anticipated, it may overwrite data still being used by the data consumer (e.g. some other component of the computer).

Additionally, as speed of execution of code by a CPU depends on the characteristics of the specific code being executed, it is possible that the degree of increase of performance of the CPUs of the new device relative to the legacy device will depend on the specific code being executed. This may lead to problems in the producer-consumer model described above, where producer and consumer are both CPUs but are executing the code of the legacy application at relative speeds not encountered on the legacy hardware.

EMBODIMENTS

Aspects of the present disclosure describe computer systems and methods which may allow for a higher degree of backwards compatibility with regards to legacy computer systems.

In implementations of the present disclosure, certain resources relating to the CPUs are restricted and various aspects of the operation of the CPUs are altered when executing in a backwards compatibility mode ("BC mode").

Due to the restriction of various resources, the performance of the CPUs in BC mode may become much closer to that of the legacy CPUs, resulting in fewer errors in legacy applications due to the unanticipated performance characteristics of the CPUs.

Additionally, certain features of the CPUs that are not present on legacy CPUs may be disabled in BC mode; the latency of instruction execution of the CPUs may be altered in BC mode to equal or become closer to the latency of legacy CPUs; and the algorithmic details of the operation of various units of the CPUs may be altered in BC mode to match or approximate the algorithmic details of the operation of those units of the legacy CPUs. As a result, when in BC mode, the performance of the new CPUs may become much closer to that of the legacy CPUs, resulting in fewer errors in legacy applications due to the unanticipated performance characteristics of the new CPUs.

The following describes the general architecture of a CPU, and various aspects of the present disclosure relating to the restriction of specific resources, disabling of features, alteration of latency, and alteration of algorithmic details of operation when in BC mode.

FIG. 1 depicts a generalized architecture of a CPU core **100**. The CPU core **100** typically includes a branch prediction unit **102**, that attempts to predict whether a branch will be taken or not, and also attempts (in the event that the branch is taken) to predict the destination address of the branch. To the extent that these predictions are correct the efficiency of speculatively executed code will be increased; highly accurate branch prediction is therefore extremely desirable. The branch prediction unit **102** may include highly specialized sub-units such as a return address stack **104** that tracks return addresses from subroutines, an indirect target array **106** that tracks the destinations of indirect branches, and a branch target buffer **108** and its associated prediction logic that track past history of branches in order to more accurately predict their resulting addresses.

According to certain aspects of the present disclosure, in BC mode the size of the indirect target array **106**, the size of the return address stack **104,** or the size of the branch target buffer **108** of the new CPU may be reduced to match, or to more closely approximate, their respective size for the legacy CPU. To be clear, this reduction takes the form of reducing the usable portion of the resource, e.g. not allowing usage of a portion of the return address stack, and thereby reducing the number of calls and associated returns that can be tracked; the full resource is available when not in BC mode.

According to certain aspects of the present disclosure, in BC mode the algorithmic details of the operation of the branch target buffer **108** of the new CPU and its associated prediction logic may be altered to match those of the legacy CPU. By way of example and not by way of limitation, if the legacy CPU is limited in its ability to track the behavior of branch instructions that are close to each other, then in BC mode the new CPU may match this legacy CPU behavior; or if the legacy CPU used a substantially different style of branch prediction logic (e.g. a saturating counter rather than an adaptive predictor) then the new CPU may include the logic of the legacy CPU and enable it in BC mode.

According to certain aspects of the present disclosure, should the branch target buffer **108** of the new CPU and its associated prediction logic include a dedicated loop predictor, but no

dedicated loop predictor is present on the legacy CPU, then in BC mode the dedicated loop predictor of the new CPU may be disabled.

The CPU core **100** typically includes an instruction fetch and decode unit **110**, which includes an instruction fetch unit **112**, an instruction byte buffer **114**, and an instruction decode unit **116**. The CPU core **100** also typically includes a number of instruction related caches and instruction translation lookaside buffers (ITLBs) **120**. These may include an ITLB cache hierarchy **124** that caches virtual address to physical address translation information such as page table entries, page directory entries, and the like. This information is used to transform the virtual address of the instruction into a physical address so that the instruction fetch unit **112** can load the instructions from the cache hierarchy. By way of example, and not by way of limitation, the program instructions may be cached according to a cache hierarchy that includes a level 1 instruction cache (L1 I-Cache) **122** residing in the core, as well as other cache levels **176** external to the CPU core **100**; using the physical address of the instruction, these caches are first searched for the program instructions. If the instructions are not found, then they are loaded from a system memory **101**. Depending on the architecture, there may also be a micro-op cache **126** that contains the decoded instructions, as described below.

In certain aspects of the present disclosure, in BC mode the size or associativity of the L1 I-cache **124**, the micro-op cache **126** or the various levels of the ITLB cache hierarchy **122** may be changed to match, or to more closely approximate, their respective size and associativity for the legacy CPU. By way of example, and not by way of limitation, changing, e.g., reducing, the size of the ITLB cache hierarchy **124** could involve (1) reducing the number of levels; or (2) changing the size of one or more levels (e.g., cache size, block size, number of blocks in a set). Altering the associativity of a cache may involve, e.g., operating a fully associative cache as a four-way or two-way cache. Although aspects of the present disclosure include implementations where a size or associativity of an instruction-related cache or ITLB is reduced, the present disclosure is not limited to such implementations. For example, it is possible for a legacy CPU has to have a larger cache that is less associative (e.g., 2-way instead of 4-way) in the legacy CPU. In such a case, the new CPU may run in BC mode with the corresponding cache size increased and associativity reduced to match or approximate the behavior of the cache on the legacy CPU.

Once the program instructions have been fetched, they are typically placed in the instruction byte buffer **114** awaiting processing by the instruction fetch and decode unit **110**. Decoding

can be a very complex process; it is difficult to decode multiple instructions each cycle, and there may be restrictions on instruction alignment or type of instruction that limit how many instructions may be decoded in a cycle. Decoded instructions may, depending on architecture, be placed in the micro-op cache **126** (if one is present on the new CPU) so that the decode stage can be bypassed for subsequent use of the program instructions.

In certain aspects of the present disclosure, in BC mode the algorithmic details of the operation of the instruction fetch and decode unit **110** of the new CPU may be altered to match those of the legacy CPU. By way of example, and not by way of limitation, if the legacy CPU restricted the decode to instructions with opcodes within a specific area in the instruction byte buffer **114**, then the new CPU may similarly restrict the decode.

In certain aspects of the present disclosure, should a micro-op cache **126** be present on the new CPU and absent on the legacy CPU, then in BC mode the micro-op cache **126** of the new CPU may be disabled.

Decoded instructions are typically passed to other units for dispatch and scheduling **130**. These units may use retirement queues **132** to track the status of the instructions throughout the remainder of the CPU pipeline. Also, due to the limited number of general purpose and SIMD registers available on many CPU architectures, register renaming may be performed, in which as logical (also known as architectural) registers are encountered in stream of instructions being executed, physical registers **140** are assigned to represent them. The physical registers **140** may include Single Instruction Multiple Data (**SIMD**) register banks **142** and General Purpose (**GP**) register banks **144**, which can be much larger in size than the number of logical registers available on the particular CPU architecture, and as a result the performance can be considerably increased. After register renaming **134** is performed, instructions are typically placed in scheduling queues **136**, from which a number of instructions may be selected each cycle (based on dependencies) for execution by execution units **150**.

In certain aspects of the present disclosure, in BC mode the size of the retirement queues **132,** the size of the scheduling queues **136,** or the size of the SIMD **142** or GP register banks **144** of the CPU may be reduced to match, or to more closely approximate, their respective size for the legacy CPU. To be clear, this reduction takes the form of reducing the usable portion of the resource, e.g. restricting the number of physical registers available to the application in

BC mode; the full register bank would be available for use by applications when not in BC mode.

The execution units **150** typically include SIMD pipes **152** that perform a number of parallel operations on multiple data fields contained in 128-bit or wider SIMD registers contained in the SIMD register bank **142**, arithmetic and logic units (ALUs) **154** that perform a number of logical, arithmetic, and miscellaneous operations on GPRs contained in the GP register bank **144**, and address generation units (AGUs) **156** that calculate the address from which memory should be stored or loaded. There may be multiple instances of each type of execution unit, and the instances may have differing capabilities, for example a specific SIMD pipe **152** may be able to perform floating point multiply operations but not floating point add operations.

In certain aspects of the present disclosure, in BC mode the usable number of ALUs, AGUs or SIMD pipes may be reduced to match, or to more closely approximate, the respective number of such units that exist on the legacy CPU.

In certain aspects of the present disclosure, in BC mode the latency of instruction execution of the new CPU may be altered in such a way as to equal or become closer to the latency of the legacy CPU; for example, in BC mode the latency of a divide operation on the new CPU could be extended (e.g. by computing the result more slowly or by delaying the transfer of the result to the subsequent stage of the pipeline) to match, or to more closely approximate, the latency of a divide operation on the legacy CPU.

Stores and loads are typically buffered in a store queue **162** and a load queue **164** so that many memory operations can be performed in parallel. To assist in memory operations, the CPU core **100** usually includes a number of data related caches and data translation lookaside buffers (DTLBs) **170**. A DTLB cache hierarchy **172** caches virtual address to physical address translation such as page table entries, page directory entries, and the like; this information is used to transform the virtual address of the memory operation into a physical address so that data can be stored or loaded from system memory. The data is typically cached in a level 1 data cache (L1 D-Cache) **174** residing in the core, as well as other cache levels **176** external to the core **100**.

In certain aspects of the present disclosure, in BC mode the size and associativity of the L1 D-cache **174** or the various levels of the DTLB cache hierarchy **172** may be reduced to match, or to more closely approximate, their respective size and associativity for the legacy

CPU. In certain aspects of the present disclosure, in BC mode the size of the store queue **162** or load queue **164** of the CPU (e.g. the number of allowable outstanding stores or loads) may be reduced to match, or to more closely approximate, their respective size for the legacy CPU.

FIG. 2 is a flow diagram illustrating an example of a possible process flow of a method in accordance with aspects of the present disclosure. The method starts at **201**, e.g., by loading an application onto a system having a new CPU. Via an examination of a software ID, software checksum, metadata associated with the software, media type, or other mechanism, a determination is made if an application is designed for a new CPU or for the prior versions of the system, as indicated at **210**. Such a determination may be implemented in software running on the system or in hardware of the system. When it is determined that the loaded application is intended for the new CPU, the system may run normally, as indicated at **220**. For example, the CPU may run normally without restriction on available resources, without disabling features, altering latency of execution of instructions, or altering algorithmic details to match or approximate behavior of a legacy CPU.

When it is determined that the loaded application is intended for a legacy CPU, the CPU runs in BC mode, with selected available resources restricted **242**, with selected features not present on the legacy CPU disabled **244**, with latency of execution of instructions altered **246**, or with algorithmic details altered **248** or some combination of two or more of these to match or approximate behavior of the legacy CPU. Examples of these possibilities are discussed above.

By way of example, and not by way of limitation to operate the CPU with selected resources restricted **242**, the BC mode may be implemented by suitable configuration of the hardware of the new CPU, by suitable configuration of an operating system that runs the CPU or some combination of both. For example, as discussed above, in BC mode the size of the indirect target array **106,** the size of the return address stack 104, or the size of the branch target buffer **108** of the CPU may be reduced to match, or to more closely approximate, their respective size for the legacy CPU. By way of example and not by way of limitation, the relevant hardware may be configured so that in BC mode the operating system or CPU firmware could reduce the size of the indirect target array **106** to match, or to more closely approximate, their respective size for the legacy CPU. The following pseudocode illustrates an example of how this might be implemented:

```
void function BC_mode_indirect_target_array_size
  if BC_mode is true {
    set indirect_target_array_size to reduced_ indirect_target_array_size
    }
```

The size of the return address stack **104,** or the size of the branch target buffer **108**, or other available resources may be reduced in similar manner.

In a like manner, to operate the CPU with selected features disabled **244** certain hardware resources present on the new CPU that are not present on the legacy CPU (e.g., the micro-op cache **126**) may be configured so that they can be disabled by the operating system or CPU firmware in BC mode. Alternatively, hardware resources present on the new CPU that are not present on the legacy CPU may be configured so that they are ignored by the application in BC mode.

By way of example, and not by way of limitation, to operate the CPU with a latency of instruction execution of the new CPUs altered **246** to match or approximate a latency of the legacy CPU the hardware of the execution units **150** may be configured to add the equivalent of "no-op" instructions in BC mode to obtain the desired latency in BC mode.

By way of example, and not by way of limitation, to operate the new CPU with algorithmic details of operation of one or more units of the new CPU altered **248**. By way of example, and not by way of limitation, the algorithmic details of operation of the branch prediction unit **102** may be altered in BC mode. For example, as discussed above, if the legacy CPU is limited in its ability to track the behavior of branch instructions that are close to each other, then in BC mode the branch prediction unit **102** may be configured to match this legacy CPU behavior in BC mode. Alternatively, if the legacy CPU uses a substantially different style of branch prediction logic (e.g. a saturating counter rather than an adaptive predictor) then the branch prediction unit **102** of the new CPU may include the logic of the legacy CPU that can be enabled in BC mode. In other implementations, the algorithmic details of operation of the instruction fetch and decode unit **110**, dispatch and scheduling unit 130, or execution units **150** of a new CPU may be similarly configured with legacy logic that could be enabled in BC mode.

Turning now to FIG. 3, an illustrative example of a system **300** configured to operate in accordance with aspects of the present disclosure is depicted. According to aspects of the

present disclosure, the system **300** may be an embedded system, mobile phone, personal computer, tablet computer, portable game device, workstation, game console, and the like.

The system **300** generally includes a central processor unit (CPU) **320** which may include a CPU core and other features of the type depicted in FIG. 1 and discussed above. By way of example and not by way of limitation, the CPU **320** may be part of an accelerated processing unit (APU) **310** that includes the CPU **320**, and a graphics processing unit (GPU) **330** on a single chip. In alternative implementations, the CPU **320** and GPU **330** may be implemented as separate hardware components on separate chips.

The system **300** may also include memory **340**. The memory **340** may optionally include a main memory unit that is accessible to the CPU **320** and GPU **330**. The CPU **320** and GPU **330** may each include one or more processor cores, e.g., a single core, two cores, four cores, eight cores, or more. The CPU **320** and GPU **330** may be configured to access one or more memory units using a data bus **390**, and, in some implementations, it may be useful for the system **300** to include two or more different buses.

The memory **340** may include one or more memory units in the form of integrated circuits that provides addressable memory, e.g., RAM, DRAM, and the like. The memory contains executable instructions configured to implement the method of FIG. 2 upon execution for determining operate the CPU **320** in a BC mode when running applications originally created for execution on a legacy CPU. In addition, the memory **340** may include a dedicated graphics memory for temporarily storing graphics resources, graphics buffers, and other graphics data for a graphics rendering pipeline.

The CPU **320** may be configured to execute CPU code, which may include operating system (OS) **321** or an application **322** (e.g., a video game). The OS **321** may be configured to implement certain features of operating the CPU **320** in the BC mode, as discussed above. The CPU code may include a graphics application programming interface (API) **324** for issuing draw commands or draw calls to programs implemented by the GPU **330** based on a state of the application **322**. The CPU code may also implement physics simulations and other functions. Portions of the code for one or more of the OS **321**, application **322**, or API **324** may be stored in the memory **340**, caches internal or external to the CPU or in a mass storage device accessible to the CPU **320**.

The system **300** may also include well-known support functions **350**, which may communicate with other components of the system, e.g., via the bus **390**. Such support functions may include, but are not limited to, input/output (I/O) elements **352**, one or more clocks **356**, which may include separate clocks for the CPU and GPU, respectively, and one or more levels of cache **358**, which may be external to the CPU **320**. The system **300** may optionally include a mass storage device **360** such as a disk drive, CD-ROM drive, flash memory, tape drive, Blu-ray drive, or the like to store programs and/or data. In one example, the mass storage device **360** may receive a computer readable medium **362** containing a legacy application originally designed to run on a system having a legacy CPU. Alternatively, the legacy application **362** (or portions thereof) may be stored in memory **340** or partly in the cache **358**.

The device **300** may also include a display unit **380** to present rendered graphics **382** prepared by the GPU **330** to a user. The device **300** may also include a user interface unit **370** to facilitate interaction between the system **100** and a user. The display unit **380** may be in the form of a flat panel display, cathode ray tube (CRT) screen, touch screen, head mounted display (HMD) or other device that can display text, numerals, graphical symbols, or images. The display **380** may display rendered graphics **382** processed in accordance with various techniques described herein. The user interface **370** may contain one or more peripherals, such as a keyboard, mouse, joystick, light pen, game controller, touch screen, and/or other device that may be used in conjunction with a graphical user interface (GUI). In certain implementations, the state of the application **322** and the underlying content of the graphics may be determined at least in part by user input through the user interface **370**, e.g., where the application **322** includes a video game or other graphics intensive application.

The system **300** may also include a network interface **372** to enable the device to communicate with other devices over a network. The network may be, e.g., a local area network (LAN), a wide area network such as the internet, a personal area network, such as a Bluetooth network or other type of network. Various ones of the components shown and described may be implemented in hardware, software, or firmware, or some combination of two or more of these.

According to aspects of the present disclosure, the CPU **320** may include hardware components such as the components of the CPU core **100** of FIG. 1 that can operate in a BC mode with selected available resources restricted **242**, with selected features not present on

the legacy CPU disabled **244**, with latency of execution of instructions altered **246**, or with algorithmic details altered **248** or some combination of two or more of these to match or approximate behavior of the legacy CPU, as discussed above with respect to FIG. 2.

Aspects of the present disclosure overcome problems with backward compatibility that arise when programs written for a legacy system run on a more powerful new system. By running the new CPU in BC mode with selected available resources restricted, with selected features not present on the legacy CPU disabled, with latency of execution of instructions altered, or with algorithmic details altered or some combination of two or more of these the new CPU can match or approximate the behavior of the legacy CPU.

While the above is a complete description of the preferred embodiment 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 described herein, whether preferred or not, may be combined with any other feature described herein, 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. As used herein, in a listing of elements in the alternative, the term "or" is used as the inclusive sense, e.g., "X or Y" covers X alone, Y alone, or both X and Y together, except where expressly stated otherwise. Two or more elements listed as alternatives may be combined together. 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."

WHAT IS CLAIMED IS:

1. A method, comprising:
with a new device executing an application on a new central processing unit (CPU), determining whether the application is for a legacy device having a legacy CPU; and when the new device determines that the application is for the legacy device, executing the application on the new CPU with selected available resources of the new device restricted to approximate or match a processing behavior of the legacy CPU.

2. The method of claim 1, wherein executing the application on the new CPU with selected available resources of the new CPU restricted includes:
reducing a size of an indirect target array of the new CPU to approximate a size of a corresponding indirect target array of the legacy CPU.

3. The method of claim 1, wherein executing the application on the new CPU with selected resources of the new CPU restricted includes reducing a size of a return address stack of the new CPU to approximate a size of a corresponding return address stack of the legacy CPU.

4. The method of claim 1, wherein executing the application on the new CPU with selected resources of the new device restricted includes reducing a size of a branch target buffer of the new CPU to approximate a size of a corresponding branch target buffer of the legacy CPU.

5. The method of claim 1, wherein executing the application on the new CPU with selected resources of the new CPU restricted includes changing a size or associativity of an instruction related cache or translation lookaside buffer of the new CPU to match or approximate a size or associativity for a corresponding instruction related cache or translation lookaside buffer of the legacy CPU.

6. The method of claim 5, wherein the instruction related cache or translation lookaside buffer of the new CPU is an instruction translation lookaside buffer cache hierarchy, a level 1 instruction cache, or a micro-op cache.

7. The method of claim 1, wherein executing the application on the new CPU with selected resources of the new CPU restricted includes reducing a size a queue of the new CPU to match or approximate a size of a corresponding queue of the legacy CPU.

8. The method of claim 7, wherein the queue of the new CPU is a retirement queue or a scheduling queue.

9. The method of claim 1, wherein executing the application on the new CPU with selected resources of the new CPU restricted includes reducing a size a register bank of the new CPU to match or approximate a corresponding size of a register bank of the legacy CPU.

10. The method of claim 9, wherein the register bank is a SIMD physical register bank or a GP physical register bank.

11. The method of claim 1, wherein executing the application on the new CPU with selected resources of the new CPU restricted includes reducing a number of usable execution units of the new CPU to match or approximate an available number of corresponding execution units of the legacy CPU.

12. The method of claim 11, wherein the execution units include Arithmetic Logic Units (ALU), Address Generation Units (AGU) or Single Instruction Multiple Data (SIMD) pipes.

13. The method of claim 1, wherein executing the application on the new CPU with selected resources of the new CPU restricted includes changing a size or associativity of cache of the new CPU to match or approximate a corresponding size or associativity of corresponding cache of the legacy CPU.

14. The method of claim 13, wherein the cache is an instruction-related cache.

15. The method of claim 13, wherein the cache is a data translation lookaside buffer (DTLB) cache hierarchy or a level 1 data cache.

16. A system, comprising,
a new central processing unit (CPU) configured to execute instructions of an application, the new CPU having logic units configured to determine whether the application is for a legacy device having a legacy CPU, and execute the application with selected available

resources of the new CPU restricted to approximate or match a processing behavior of the legacy CPU when the application is for the legacy device.

17. The system of claim 16, wherein the selected available resources of the new CPU are restricted by reducing a size of an indirect target array of the new CPU to approximate a size of a corresponding indirect target array of the legacy CPU.

18. The system of claim 16, wherein the selected available resources of the new CPU are restricted by reducing a size of a return address stack of the new CPU to approximate a size of a corresponding return address stack of the legacy CPU.

19. The system of claim 16, wherein the selected available resources of the new CPU are restricted by reducing a size of a branch target buffer of the new CPU to approximate a size of a corresponding branch target buffer of the legacy CPU.

20. The system of claim 16, wherein the selected available resources of the new CPU are restricted by changing a size or associativity of an instruction related cache or translation lookaside buffer of the new CPU to match or approximate a size or associativity for a corresponding instruction related cache or translation lookaside buffer of the legacy CPU.

21. The system of claim 20, wherein the instruction related cache or translation lookaside buffer of the new CPU is an instruction translation lookaside buffer cache hierarchy, a level 1 instruction cache, or a micro-op cache.

22. The system of claim 16, wherein the selected available resources of the new CPU are restricted by reducing a size a queue of the new CPU to match or approximate a size of a corresponding queue of the legacy CPU.

23. The system of claim 22, wherein the queue of the new CPU is a retirement queue or a scheduling queue.

24. The system of claim 16, wherein the selected available resources of the new CPU are restricted by reducing a size a register bank of the new CPU to match or approximate a corresponding size of a register bank of the legacy CPU.

25. The system of claim 24, wherein the register bank is a SIMD physical register bank or a GP physical register bank.

26. The system of claim 16, wherein the selected available resources of the new CPU are restricted by reducing a number of usable execution units of the new CPU to match or approximate an available number of corresponding execution units of the legacy CPU.

27. The system of claim 26, wherein the execution units include Arithmetic Logic Units (ALU), Address Generation Units (AGU) or Single Instruction Multiple Data (SIMD) pipes.

28. The system of claim 16, wherein the selected available resources of the new CPU restricted are restricted by changing a size or associativity of cache of the new CPU to match or approximate a corresponding size or associativity of corresponding cache of the legacy CPU.

29. The system of claim 28, wherein the cache is an instruction-related cache.

30. The system of claim 28, wherein the cache is a data translation lookaside buffer (DTLB) cache hierarchy or a level 1 data cache.

31. A non-transitory computer readable medium having executable instructions embodied therein, the instructions being configured to implement a method upon execution of the instructions, the method comprising:
with a new device executing an application on a new central processing unit (CPU), determining whether the application is for a legacy device having a legacy CPU; and
when the new device determines that the application is for the legacy device executing the application on the new CPU with selected available resources of the new device restricted to approximate or match a processing behavior of the legacy CPU.

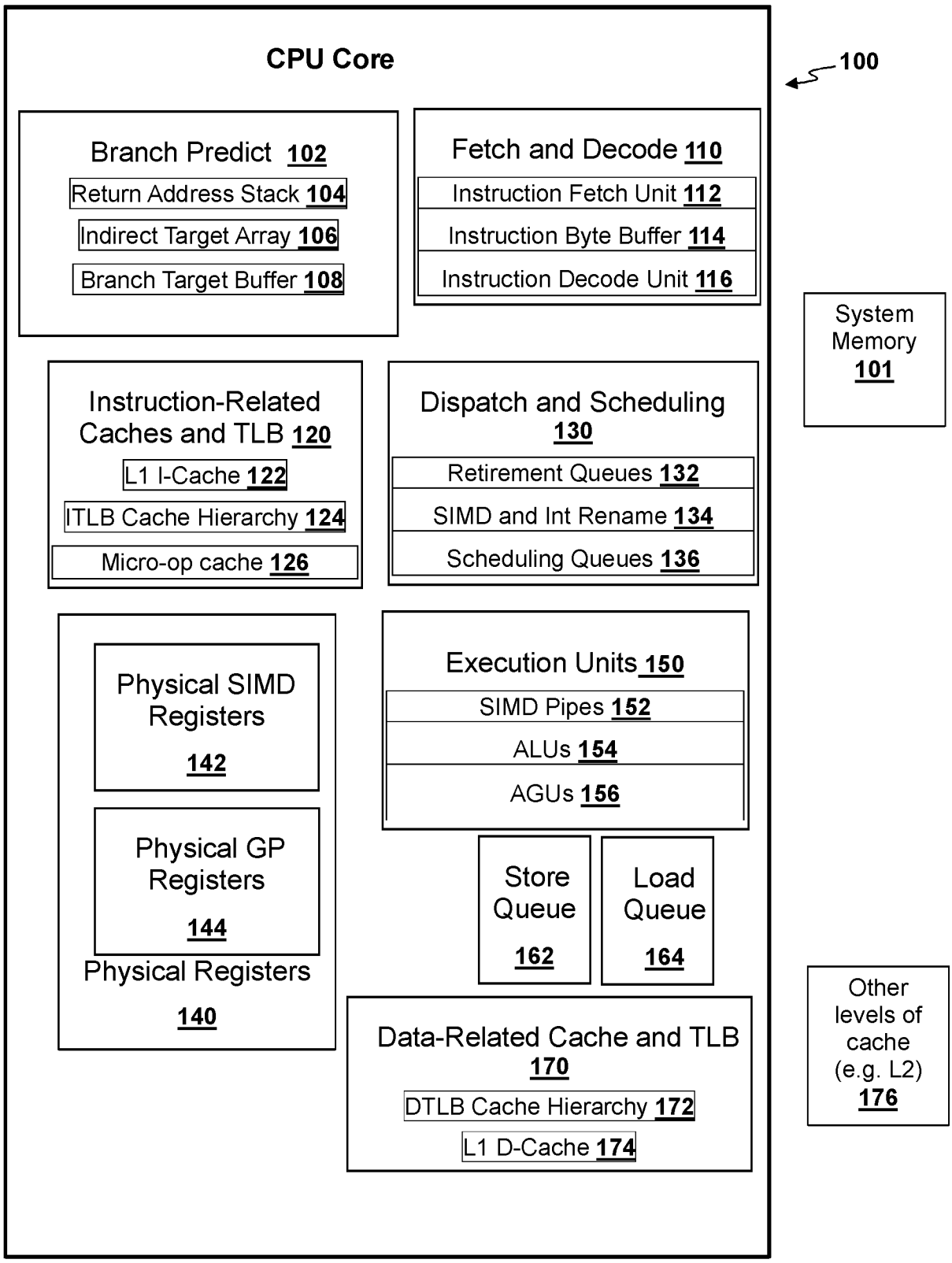
CPU Core
100
Branch Predict 102
Return Address Stack 104
Indirect Target Array 106
Branch Target Buffer 108
Fetch and Decode 110
Instruction Fetch Unit 112
Instruction Byte Buffer 114
Instruction Decode Unit 116
System Memory 101
Instruction-Related Caches and TLB 120
L1 I-Cache 122
ITLB Cache Hierarchy 124
Micro-op cache 126
Dispatch and Scheduling 130
Retirement Queues 132
SIMD and Int Rename 134
Scheduling Queues 136
Physical SIMD Registers 142
Physical GP Registers 144
Physical Registers 140
Execution Units 150
SIMD Pipes 152
ALUs 154
AGUs 156
Store Queue 162
Load Queue 164
Data-Related Cache and TLB 170
DTLB Cache Hierarchy 172
L1 D-Cache 174
Other levels of cache (e.g. L2) 176

FIG. 1

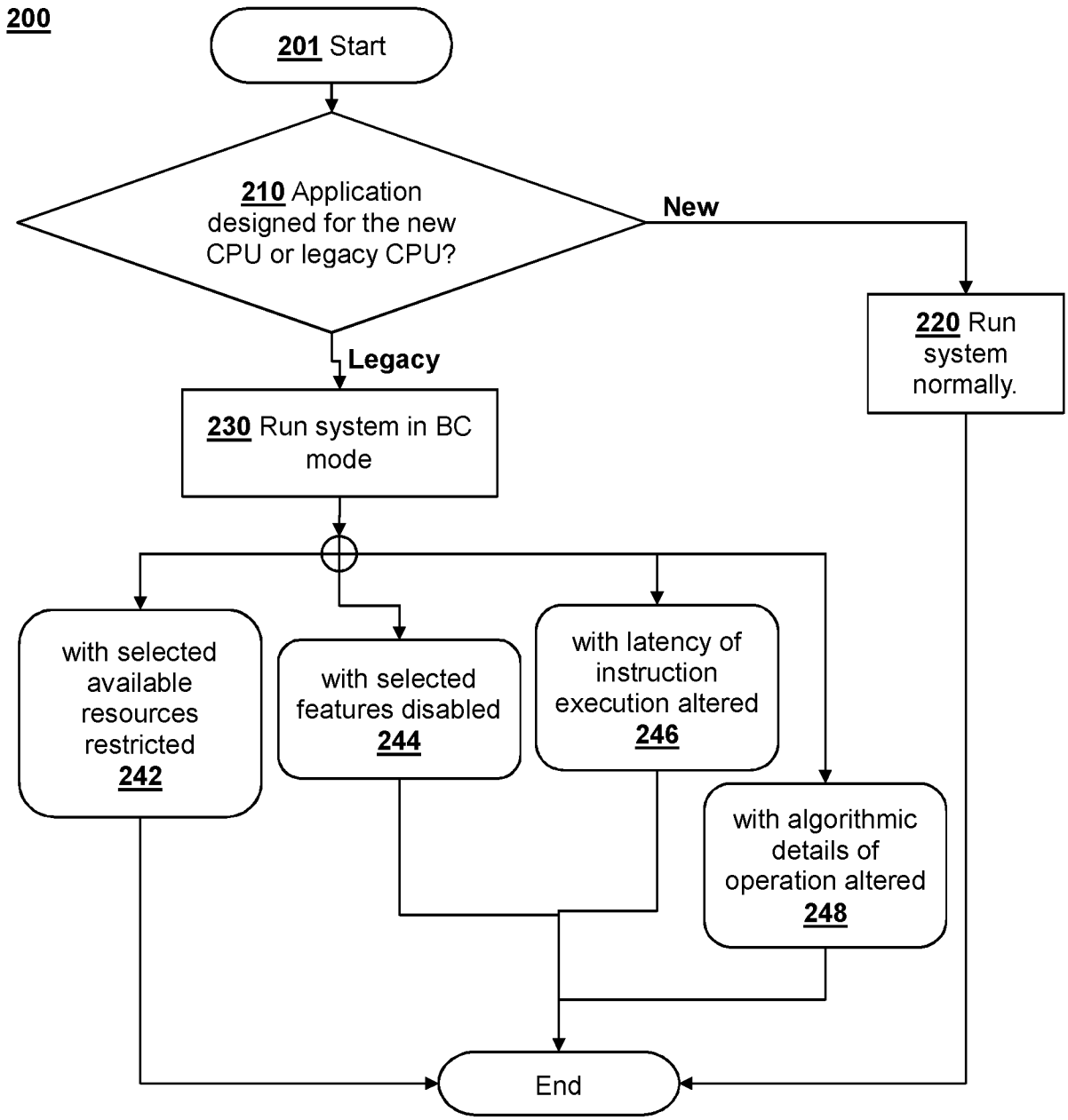
200
201 Start
210 Application designed for the new CPU or legacy CPU?
New
220 Run system normally.
Legacy
230 Run system in BC mode
with selected available resources restricted 242
with selected features disabled 244
with latency of instruction execution altered 246
with algorithmic details of operation altered 248
End

FIG. 2

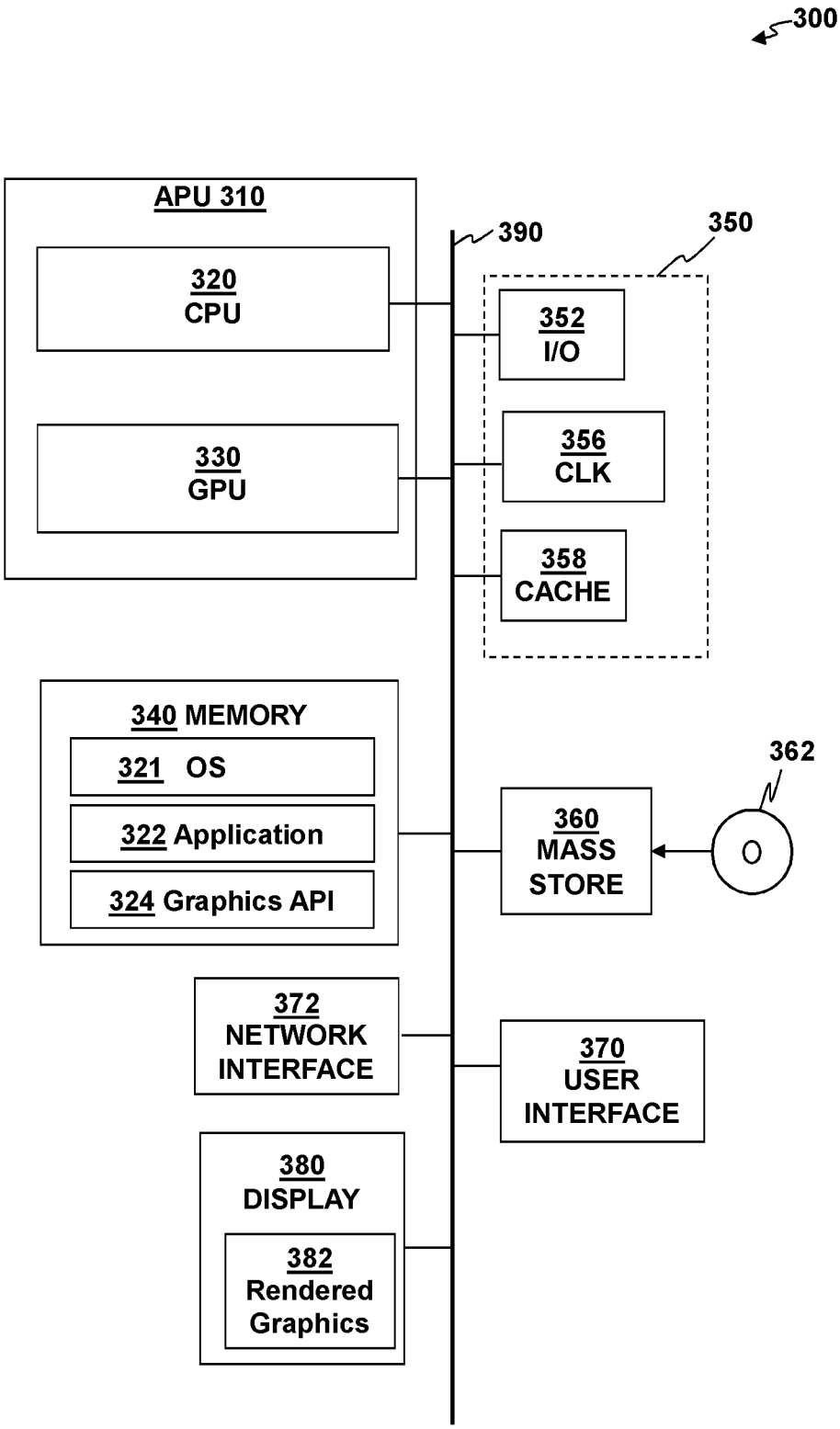
300
APU 310
320
CPU
330
GPU
390
350
352
I/O
356
CLK
358
CACHE
340 MEMORY
321 OS
322 Application
324 Graphics API
362
360
MASS
STORE
372
NETWORK
INTERFACE
370
USER
INTERFACE
380
DISPLAY
382
Rendered
Graphics

FIG. 3

INTERNATIONAL SEARCH REPORT

International application No.
PCT/US 16/41762

A. CLASSIFICATION OF SUBJECT MATTER
IPC(8) - G06F 9/44 (2016.01)
CPC - G06F8/71
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)
USPC: 717/122; CPC: G06F8/71; IPC(8): G06F 9/44 (2016.01)

Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched
USPC: 717/122; CPC: G06F8/71, G06F9/44521, G06F8/30, G06F8/36, G06F8/66; IPC(8): G06F 9/44 (2016.01) (keyword limited, terms below)

Electronic data base consulted during the international search (name of data base and, where practicable, search terms used)
PatBase, Google Patents, IEEE; Search Terms: legacy device; legacy CPU, processor; translation lookaside buffer, TLB, cache; data translation lookaside buffer (DTLB); hierarchy; emulation, simulation; reducing, restricting, re-sizing; resources; indirect target array; branch target buffer; retirement, scheduling queue; execution unit; register ba

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 2005/0033831 A1 (Rashid) 10 February 2005 (10.02.2005), entire document especially paras [0064], [0066]	1, 16, 31 ---------- 2-15, 17-30
Y	US 2014/0282425 A1 (Zhao et al.) 18 September 2014 (18.09.2014), entire document especially para [0033]	2-15, 17-30
Y	US 2013/0339649 A1 (Hsu et al.) 19 December 2013 (19.12.2013), entire document especially Abstract, paras [0034], [0042]	10, 25
A	US 2009/0119477 A1 (Plondke et al.) 07 May 2009 (07.05.2009), entire document	1-31
A	US 6,772,315 B1 (Perego) 03 August 2004 (03.08.2004), entire document	1-31
A	US 2014/0304771 A1 (Reierson et al.) 09 October 2014 (09.10.2014), entire document	1-31
A	US 2009/0063772 A1 (Magoshi) 05 March 2009 (05.03.2009), entire document	1-31

☐ 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	Date of mailing of the international search report
01 September 2016 (01.09.2016)	05 OCT 2016

Name and mailing address of the ISA/US	Authorized officer:
Mail Stop PCT, Attn: ISA/US, Commissioner for Patents P.O. Box 1450, Alexandria, Virginia 22313-1450 Facsimile No. 571-273-8300	Lee W. Young PCT Helpdesk: 571-272-4300 PCT OSP: 571-272-7774