(54) Title: PROVIDING LOGICAL PARTITIONS WITH HARDWARE-THREAD SPECIFIC INFORMATION REFLECTIVE OF EXCLUSIVE USE OF A PROCESSOR CORE
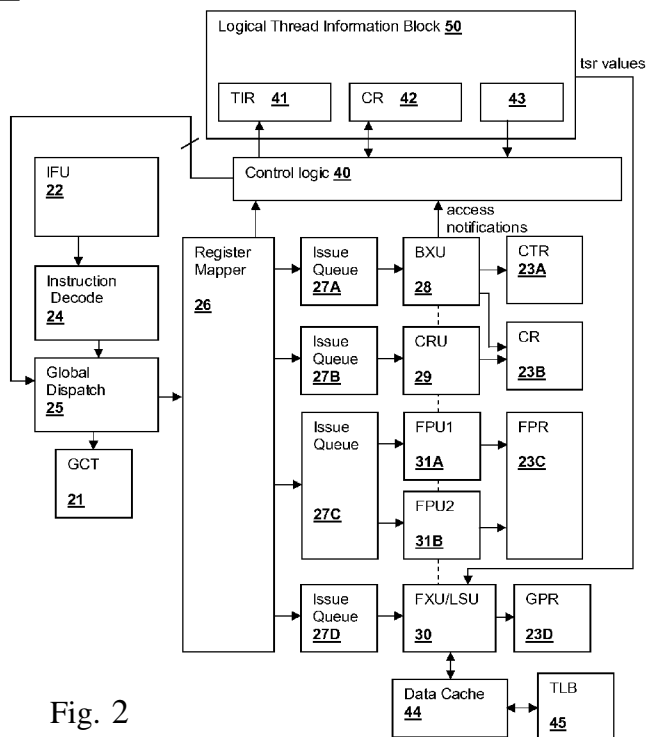


Fig. 2

(57) Abstract: Techniques for simulating exclusive use of a processor core amongst multiple logical partitions (LPARs) include providing hardware thread-dependent status information in response to access requests by the LPARs that is reflective of exclusive use of the processor by the LPAR accessing the hardware thread-dependent information. The information returned in response to the access requests is transformed if the requestor is a program executing at a privilege level lower than the hypervisor privilege level, so that each logical partition views the processor as though it has exclusive use of the processor. The techniques may be implemented by a logical circuit block within the processor core that transforms the hardware thread-specific information to a logical representation of the hardware thread- specific information or the transformation may be performed by program instructions of an interrupt handler that traps access to the physical register containing the information.

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, ML, MR, NE, SN, TD, TG).

**Published:**

— *without international search report and to be republished upon receipt of that report (Rule 48.2(g))*

# PROVIDING LOGICAL PARTITIONS WITH HARDWARE-THREAD SPECIFIC INFORMATION REFLECTIVE OF EXCLUSIVE USE OF A PROCESSOR CORE

## BACKGROUND OF THE INVENTION

### Field of the Invention

[0001] The present invention is related to processing systems and processors, and more specifically to techniques for managing values of hardware thread specific information returned to logical partitions executed by hardware threads of a processor core.

### Description of Related Art

[0002] In large scale computer systems, in particular in multi-user computer systems or so-called cloud computing systems in which multiple processors support multiple virtual operating systems and images, referred to as logical partitions (LPARs), a hypervisor manages allocation of resources to the LPARs, and the starting/stopping of LPARs at system startup/shutdown and context swaps. The LPARs are preferably isolated from each other when the system is used, for example, to support execution of different LPARs for different customers. Typically, the processor cores used in such a computer system support concurrent execution of multiple hardware threads, e.g., 8 or more hardware threads, which correspond to parallel execution pipelines within the processor core. Typically also, the hypervisor assigns a number of threads to each LPAR.

[0003] Status and other informational values within the processor are hardware thread-specific, i.e., the values may be either a value for each hardware thread number, be dependent on the thread trying to access the value, or may be the number of hardware threads that the processor supports.

[0004] It would be desirable to provide techniques for managing accesses to hardware thread-specific information that are independent of which LPAR is accessing the information and provides security between LPARS.

## BRIEF SUMMARY OF THE INVENTION

[0005] The invention is embodied in a method, computer systems, processor core and computer program product that provide hardware thread-dependent status information in response to access requests. The information returned in response to the access requests is transformed if the requestor is a program executing at a privilege level lower than the hypervisor privilege level, so that each logical partition views the processor as though it has exclusive use of the processor core.

[0006] The method may be implemented by a logical circuit block within the processor that transforms the hardware thread-specific information to a logical representation of the hardware thread-specific information that reflects exclusive use of the processor core by threads of a given logical partition executed by the processor. Alternatively, the transformation may be performed by program instructions of an interrupt handler that traps access to the physical register of the processor core that contains the information. The interrupt handler either transforms the contents of the register directly, or performs the transformation via a look-up in memory of a table that contains previously transformed thread-specific information.

[0007] The foregoing and other objectives, features, and advantages of the invention will be apparent from the following, more particular, description of the preferred embodiment of the invention, as illustrated in the accompanying drawings.

## BRIEF DESCRIPTION OF THE SEVERAL VIEWS OF THE DRAWING

[0008] The novel features believed characteristic of the invention are set forth in the appended claims. The invention itself, however, as well as a preferred mode of use, further objectives, and advantages thereof, will best be understood by reference to the following detailed description of the invention when read in conjunction with the accompanying Figures, wherein like reference numerals indicate like components, and:

[0009] Figure 1 is a block diagram illustrating a processing system in which techniques according to an embodiment of the present invention are practiced.

[0010] Figure 2 is a block diagram illustrating details of a processor core 20 in accordance with an embodiment of the present invention.

[001 1] Figure 3 is a block diagram illustrating details of logical thread information block 50 within processor core 20 of Figure 2, in accordance with an embodiment of the present invention.

[0012] Figure 4 is a flowchart depicting a method in accordance with another embodiment of the present invention.

## DETAILED DESCRIPTION OF THE INVENTION

[0013] The present invention relates to processor cores and processing systems in which hardware thread-dependent stored within processor cores is transformed when accessed by logical partitions (LPARs) that access the information. The transformation is performed such that it appears to each LPAR that the LPAR has exclusive use of the processor core. For example, if a processor core supports 8 hardware threads and the threads are apportioned equally between two LPARs being executed by the processor core, then requests for the number of hardware threads supported by the processor will be returned as 4 when the LPAR accesses that information. Similarly, arrays of hardware thread-specific information or words containing fields of thread-specific information will be modified to contain only 4 entries commencing at zero (or other base thread number used by the processor). Transformation of the hardware thread-specific information is triggered by an access from a privilege level lower than the hypervisor privilege level, so that when the hypervisor accesses the hardware thread-specific information, the hardware thread-specific information for all hardware threads is returned. The transformation of hardware-thread specific register values can be performed by a logical circuit, or by an interrupt handler that traps access to the register and returns the transformed value, which may be retrieved from a table in memory, or computed directly by the interrupt handler.

[0014] Referring now to Figure 1, a processing system in accordance with an embodiment of the present invention is shown. The depicted processing system includes a number of processors 10A-10D, each in conformity with an embodiment of the present invention. The depicted multi-processing system is illustrative, and processing system in accordance with other embodiments of the present invention includes uni-processor systems having symmetric multi-threading (SMT) cores. Processors 10A-10D are identical in structure and include cores 20A-20B and local storage 12, which may be a cache level, or a level of internal system memory. Processors 10A-10B are coupled to main system memory 14, a storage subsystem 16, which includes non-removable drives and optical drives, for reading media such as a CD-ROM 17 forming a computer program product and containing program instructions implementing a hypervisor for controlling multiple logical partitions (LPAR) for execution by processors 10A-10D, and also operating systems each of which has at least one supervisory thread for managing the operating system scheduler and other services, and also containing executable programs that implement applications and services executing within the logical partitions. The illustrated processing system also includes input/output (I/O) interfaces and devices 18 such as mice and keyboards for receiving user input and graphical displays for displaying information. While the system of Figure 1 is used to provide an illustration of a system in which the processor architecture of the present invention is implemented, it is understood that the depicted architecture is not limiting and is intended to provide an example of a suitable computer system in which the techniques of the present invention are applied.

[0015] Referring now to Figure 2, details of processor cores 20A-20B of Figure 1 are illustrated in depicted core 20. Core 20 includes an instruction fetch unit (IFU) 22 that fetches one or more instruction streams from cache or system memory and presents the instruction stream(s) to an instruction decode unit 24. A global dispatch unit 25 dispatches the decoded instructions to a number of internal processor pipelines. The processor pipelines each include a register mapper 26, one of issue queues 27A-27D, and an execution unit provided by branch execution unit (BXU) 28, condition result unit (CRU) 29, fixed-point unit load-store unit (FXU/LSU) 30 or floating point units (FPUs) 31A-31B. Registers such as counter register (CTR) 23A, condition register (CR) 23B general-purpose registers (GPR) 23D, and floating-point result registers (FPR) 23C provide locations for results of operations

performed by the corresponding execution unit(s). A global completion table (GCT) 21 provides an indication of pending operations that is marked as completed when the results of an instruction are transferred to the corresponding one of result registers 23A-23D. Register mappers 26 allocate storage in the various register sets so that concurrent execution of program code can be supported by the various pipelines. FXU/LSU 30 is coupled to a data cache 44 that provides for loading and storing of data values in memory that are needed or modified by the pipelines in core 20. Data cache 44 is coupled to one or more translation look-aside buffers (TLB) 45 that map real or virtual addresses in data cache 44 to addresses in an external memory space.

[0016] Exemplary processor core 20 also includes a logical thread information block 50 that, in the illustrative example and for convenience of illustration, contains all of the hardware-thread specific registers within processor core 20. While the hardware-thread specific registers may generally be co-located with other registers or in any location, in the depicted embodiment of processor core, they are organized with other logic that controls access by FXU/LSU 30 to the hardware thread-dependent status information within core 20, by both the hypervisor and by the LPARs and applications executed within the LPARs. Exemplary hardware thread-dependent status information includes thread identification register (TIR) 41 and thread status register (CR) 42, along with an inter-thread message delivery status register 43. Table I below lists the exemplary hardware thread-specific status registers and their function and physical register contents/ranges. Table I, also shows the logical ranges/contents that will be returned to an LPAR that attempts to access the physical hardware thread-specific register, either by the control logic described below with reference to Figure 3 or by the interrupt handling software described below with reference to Figure 4.

| Register Name | Function/Meaning | Physical Value range | Logical Value Range | Logical Value Transform algorithm |
|---|---|---|---|---|
| TIR (Thread Identification Register) | Contains the thread number of the accessing process | 0-N where N = number of threads minus 1 | 0-M, where M = number of threads allocated to the LPAR minus 1 | TIR - base[LPAR] |
| Thread Execution Status Register (CR) | Indicates whether a thread is active (executing) | Contains a bit for each of threads 0-N | Contains left-aligned bits for each of threads 0-M | shl( base[LPAR]) & mask |
| Inter-thread message delivery status registers | Indicates whether messages have been delivered to target thread | Contains a bit for each of threads 0-N | Contains left-aligned bits for each of threads 0-M | shl(base[LPAR]) & mask |

**Table I**

TIR 41 is a register that returns the value of the hardware thread number of the thread that is accessing TIR 41. In order to provide a logical equivalent to the LPAR that simulates exclusive use of processor core 20 by a hardware thread, the thread number must be normalized to range of hardware thread numbers that belong to the LPAR. For simplicity and the sake of illustration, it is assumed herein that the threads allocated for executing an LPAR are consecutively numbered and extend from a base hardware thread number: base[LPAR] to base[LPAR]+M, where M is one less than the number of threads allocated for the LPAR.

[0017] Transforming the value of TIR 41 merely requires subtracting base[LPAR] from the thread number in TIR 41. As an example, for a case in which LPAR #1 is allocated

hardware threads 0-2 and thus has base[LPAR]=0, and LPAR #2 is allocated hardware threads 3-7, and thus has base[LPAR]=3, when physical thread 5 (of LPAR #2) accesses TIR 41, the logical value that should be returned by hardware or software implementing the present invention is 2 (i.e., physical thread 5 minus base[LPAR], which is 2). Thus the transformation of the physical thread-specific information to logical thread-specific information makes it appear to LPAR # 2 that the range of TIR 4 1 values extends from 0 through 4, which would be the result of exclusive use of a processor core that supports five hardware threads. CR 42 is a register that contains the execution state (live/dead) of each hardware thread, with a bit for each hardware thread supported by processor core 20. Assuming that all bits are numbered from left to right starting at zero, when an LPAR accesses CR 42, the value returned contains a bit for each of the threads allocated for the LPAR, so the physical value of CR 42 is shifted left by base[LPAR] bit positions to obtain the logical value of CR 42 returned to the LPAR and is further masked with a mask value that contains a 1 bit for each hardware thread allocated to the LPAR. So, for the example given above, CR 42, when accessed by a thread of LPAR #2, which has a base[LPAR] = 3, would be shifted left by three bit positions and masked. The logical value of CR 42 for LPAR #2 is a 5 bit field that contains bits 3 through 7 of CR 42 that is masked with mask 11111000 to remove information that may have been shifted in from the right. LPAR #1, which has base[LPAR] = 0 receives a 3 bit field that contains only bits 0 through 2 of CR 42, which is the un-shifted value of CR 42 (since base[LPAR] = 0) masked with mask = 11100000 to remove bits corresponding to physical threads 3-7. Transforming the value of inter-thread message delivery status register 43 is the same as the transformation for control register CR 42 described above. Inter-thread message delivery status register 43 contains bits that indicate whether a message has been delivered to another thread. The register contains a bit for each of the other threads, so that messages to each other thread from the thread accessing the register can be tracked. In order to transform the value of inter-thread message delivery status register 43, the actual value of inter-thread message delivery status register 43 is shifted left by the number of bits specified by base[LPAR]. Thus, in the example given above, LPAR #2 receives a value for inter-thread message delivery status register 43 that is a 5-bit field containing the bits for threads 3-7, which is shifted left by 3 bits and masked with a value of 11111000 to remove information that may have been shifted in from the right. Similarly, LPAR #1 receives a value for inter-thread message delivery

8

status register 43 that is a 3-bit field containing the bits for threads 0-2, which is not shifted and is masked with a value of 11100000 to remove information about physical threads 3-7. When sending messages between threads, the program instructions executed by the LPARs use logical thread numbers. If a logical thread number to which a message is sent is outside of the range of thread numbers allocated for executing the LPAR originating the message, an error is reported. Messaging to threads that are not allocated for executing the LPAR are thus blocked to prevent communication with a thread of another LPAR.

[0019]  Referring now to Figure 3, details of logical thread information block 50 of Figure 2 are shown, in accordance with an embodiment of the present invention. While there are many different ways to implement logical thread information block 50, the circuit depicted in Figure 3 provides one example. Within logical thread information block 50, hardware thread-dependent information status registers are represented by a register 51 and it is understood that the inputs outputs and circuits required to perform the transformations described above will be duplicated as needed for each such status register 51. A selector 53 selects the output of status register 51 except when status register 51 is being accessed by a non-hypervisor process ( control signal /hyp = 1). Logical-AND gate        54 provides the selection signal to selector 53, which, when status register 51 is accessed by a non-hypervisor process, selects the output of transform logic 52, which performs the transformations described above with reference to Figure 2.

[0020]  Referring now to Figure 4, a method of providing hardware thread-dependent status information to LPARs is illustrated in a flowchart, in accordance with an alternative embodiment of the present invention. The method illustrated in Figure 4 does not require implementation of logical thread information block 50 in processor core 20, and provides an alternative mechanism for transforming the physical hardware thread-dependent register values to values that simulate exclusive use of processor core 20 by each of the LPARs. As illustrated in Figure 4, when a program accesses a register containing hardware-thread specific information (step 60), if the accessing process is the hypervisor (decision 61), the physical register value is returned (step 62). However, if the accessing process is not the hypervisor (decision 61), i.e., the accessing process belongs to one of the LPARs, a hardware interrupt is triggered to trap the access (step 63). Processor core 20 has programmatic control

of the return address to which the interrupt service handler will return and sets the return address to the address of the instruction following the instruction that attempted to access the hardware thread-dependent register. The logical register value is obtained by either programmatically performing operations similar to those performed by the logic described above with reference to Figure 3, or, the transformation can be performed in advance and maintained every time one of the thread-dependent status information registers is updated in memory, so that the logical hardware thread-dependent register values can be retrieved (step 64) and then returned in response to the access (step 65), which is generally performed by writing the target register or memory location specified by the accessing instruction with the transformed hardware thread dependent information. Until the system is shutdown or optionally if an LPAR actually does have full and exclusive use of the processor core (decision 65), the process of steps 60-65 is repeated.

[0021] As noted above, portions of the present invention may be embodied in a computer program product, which may include firmware, an image in system memory or another memory/cache, or stored on a fixed or re-writable media such as an optical disc having computer-readable code stored thereon. Any combination of one or more computer readable medium(s) may store a program in accordance with an embodiment of the invention. The computer readable medium may be a computer readable signal medium or a computer readable storage medium. A computer readable storage medium may be, for example, but not limited to, an electronic, magnetic, optical, electromagnetic, infrared, or semiconductor system, apparatus, or device, or any suitable combination of the foregoing. More specific examples (a non-exhaustive list) of the computer readable storage medium would include the following: an electrical connection having one or more wires, a portable computer diskette, a hard disk, a random access memory (RAM), a read-only memory (ROM), an erasable programmable read-only memory (EPROM or Flash memory), an optical fiber, a portable compact disc read-only memory (CD-ROM), an optical storage device, a magnetic storage device, or any suitable combination of the foregoing.

[0022] In the context of the present application, a computer readable storage medium may be any tangible medium that can contain, or store a program for use by or in connection with an instruction execution system, apparatus, or device. A computer readable signal medium

may include a propagated data signal with computer readable program code embodied therein, for example, in baseband or as part of a carrier wave. Such a propagated signal may take any of a variety of forms, including, but not limited to, electro-magnetic, optical, or any suitable combination thereof. A computer readable signal medium may be any computer readable medium that is not a computer readable storage medium and that can communicate, propagate, or transport a program for use by or in connection with an instruction execution system, apparatus, or device. Program code embodied on a computer readable medium may be transmitted using any appropriate medium, including but not limited to wireless, wireline, optical fiber cable, RF, etc., or any suitable combination of the foregoing.

[0023] While the invention has been particularly shown and described with reference to the preferred embodiments thereof, it will be understood by those skilled in the art that the foregoing and other changes in form, and details may be made therein without departing from the spirit and scope of the invention.

## CLAIMS

1.      A method of providing hardware thread-specific status information in a multi-threaded processor executing multiple logical partitions, the method comprising:

maintaining the hardware thread specific status information in a register within the multi-threaded processor;

detecting access to the hardware thread-specific status information by a currently-executing process;

determining a privilege level of the currently-executing process; and

responsive to determining that the privilege level of the currently-executing process corresponds to a process within a given one of the multiple logical partitions to which less than all of the hardware threads supported by the processor are assigned, transforming the hardware thread-specific status information in a manner that appears as exclusive use of the processor by threads of the given logical partition, while a number of hardware threads of the processor are assigned to another logical partition other than the given logical partition.

2.      The method of Claim 1, further comprising responsive to determining that the privilege level of the currently-executing process corresponds to a hypervisor that manages the multiple logical partitions, formatting the hardware thread-specific status information in a manner that shows all of the hardware thread-specific status information for all hardware threads supported by the processor.

3.      The method of Claim 1, wherein the processor maintains the hardware thread-specific status information for all hardware threads supported by the processor in a register within the processor, and wherein the transforming is performed by a circuit comprising:

a logic block that generates a logical register value from the hardware thread-specific information in the register to simulate the exclusive use of the processor by the given partition, wherein the logical register value is different than the actual value in the register; and

a data selector coupled to the logic block that selects an output of the logic block to provide the logical register value to the currently-executing process in response to the access.

4.      The method of Claim 1, wherein the processor maintains the hardware thread-specific status information for all hardware threads supported by the processor in a register within the processor, and wherein the formatting is performed by:

generating a trap interrupt when the access to the hardware thread-specific information is detected by the detecting;

transforming, by the processor executing an interrupt service routine that services the trap interrupt, the hardware thread-specific information in the register to simulate the exclusive use of the processor by the given partition to generate a logical register value that is different than the actual value in the register; and

replacing a return value for the access to the hardware thread-specific information with the logical register value.

5.      The method of Claim 1, wherein the hardware thread-specific status information is a number of hardware threads supported by the processor, and wherein the transforming returns the number of hardware threads assigned to the given partition.

6.      The method of Claim 1, wherein the hardware thread-specific status information is a value dependent on a logical hardware thread number of a particular hardware thread assigned to the given partition and specified by the access, and wherein the transforming comprises:

altering the logical hardware thread number to determine the actual hardware thread number;

retrieving the hardware thread-specific information for the actual hardware thread; and

returning the retrieved hardware thread-specific information in response to the access.

7.      The method of Claim 6, wherein the hardware thread-specific status information is a hardware thread number of the hardware thread performing the access, and wherein the altering subtracts a base thread number for the given partition to obtain the actual hardware thread number.

8.     A processor core, comprising:

one or more execution units for executing instructions of multiple hardware threads;

an instruction fetch unit for fetching the instructions;

an instruction dispatch unit for dispatching the instructions to the execution units;

a first register containing hardware thread-specific information;

a logic circuit for transforming the hardware thread-specific information to a logical representation of the hardware thread-specific information to reflect exclusive use of the processor by threads of a given logical partition executed by the processor, while a number of hardware threads of the processor are assigned to another logical partition other than the given logical partition.

9.     The processor core of Claim 8, wherein an output of the logic circuit is selected when the first register is accessed at a privilege level lower than a hypervisor privilege level.

10.     The processor core of Claim 9, further comprising a data selector for selecting between the output of the logic circuit when the first register is accessed from the privilege level lower than the hypervisor privilege level and a value in the first register when the first register is accessed from the hypervisor privilege level.

11.     The processor core of Claim 8, wherein the hardware thread-specific status information is a number of hardware threads supported by the processor, and wherein the logical representation of the hardware thread-specific information is a number of hardware threads assigned to the given partition.

12.     The processor core of Claim 8, wherein the hardware thread-specific status information is a value dependent on a logical hardware thread number of a particular hardware thread assigned to the given partition and specified when accessing the first register, and wherein the logic circuit alters the logical hardware thread number to determine the actual hardware thread number, retrieves the hardware thread-specific information for the actual hardware thread, and provides the retrieved hardware thread-specific information as an output.

13.     The processor core of Claim 12, wherein the hardware thread-specific status information is a hardware thread number of the hardware thread performing the access, and wherein the logic circuit subtracts a base thread number for the given partition to obtain the actual hardware thread number.

14.     A computer system comprising:

a memory for storing program instructions and data values; and

a processor coupled to the memory for executing the program instructions, wherein the processor comprises one or more execution units for executing instructions of multiple hardware threads, an instruction fetch unit for fetching the instructions, an instruction dispatch unit for dispatching the instructions to the execution units, a first register containing hardware thread-specific information, a logic circuit for transforming the hardware thread-specific information to a logical representation of the hardware thread-specific information to reflect exclusive use of the processor by threads of a given logical partition executed by the processor, while a number of hardware threads of the processor are assigned to another logical partition other than the given logical partition.

15.     The computer system of Claim 14, wherein an output of the logic circuit is selected when the first register is accessed at a privilege level lower than a hypervisor privilege level.

16.     The computer system of Claim 15, wherein the processor further comprises a data selector for selecting between the output of the logic circuit when the first register is accessed from the privilege level lower than the hypervisor privilege level and a value in the first register when the first register is accessed from the hypervisor privilege level.

17.     The computer system of Claim 14, wherein the hardware thread-specific status information is a number of hardware threads supported by the processor, and wherein the logical representation of the hardware thread-specific information is a number of hardware threads assigned to the given partition.

18.     The computer system of Claim 14, wherein the hardware thread-specific status information is a value dependent on a logical hardware thread number of a particular

hardware thread assigned to the given partition and specified when accessing the first register, and wherein the logic circuit alters the logical hardware thread number to determine the actual hardware thread number, retrieves the hardware thread-specific information for the actual hardware thread, and provides the retrieved hardware thread-specific information as an output.

19. The computer system of Claim 18, wherein the hardware thread-specific status information is a hardware thread number of the hardware thread performing the access, and wherein the logic circuit subtracts a base thread number for the given partition to obtain the actual hardware thread number.

20. A computer program product comprising a computer-readable storage medium storing program instructions for execution by a computer system, wherein the program instructions are program instructions of a hypervisor for managing multiple logical partitions executing within a processor core, wherein the program instructions comprise:

program instructions of an interrupt handler to, responsive to a trap interrupt generated upon access to a register containing hardware-thread specific information, transform the hardware thread-specific information in the register to simulate the exclusive use of the processor core by a given partition to generate a logical register value that is different than the actual value in the register; and

program instructions of the interrupt handler to replace a return value for the access to the hardware thread-specific information with the logical register value.

21. The computer program product of Claim 20, wherein the hardware thread-specific status information is a number of hardware threads supported by the processor core, and wherein the program instructions that transform return the number of hardware threads assigned to the given partition.

22. The computer program product of Claim 20, wherein the hardware thread-specific status information is a value dependent on a logical hardware thread number of a particular hardware thread assigned to the given partition and specified by the access, and wherein the program instructions that transform comprise program instructions for:

altering the logical hardware thread number to determine the actual hardware thread number;

retrieving the hardware thread-specific information for the actual hardware thread; and

returning the retrieved hardware thread-specific information in response to the access.

23.     The computer program product of Claim 22, wherein the hardware thread-specific status information is a hardware thread number of the hardware thread performing the access, and wherein the altering subtracts a base thread number for the given partition to obtain the actual hardware thread number.

24.     A computer system comprising:

a memory for storing program instructions and data values; and

a processor coupled to the memory for executing the program instructions, wherein the program instructions are program instructions of a hypervisor for managing multiple logical partitions executing within the processor, wherein the program instructions comprise program instructions of an interrupt handler to, responsive to a trap interrupt generated upon access to a register containing hardware-thread specific information, transform the hardware thread-specific information in the register to simulate the exclusive use of the processor core by a given partition to generate a logical register value that is different than the actual value in the register, and program instructions of the interrupt handler to replace a return value for the access to the hardware thread-specific information with the logical register value.

25.     The computer system of Claim 24, wherein the hardware thread-specific status information is a number of hardware threads supported by the processor, and wherein the program instructions that transform return the number of hardware threads assigned to the given partition.

26.     The computer system of Claim 24, wherein the hardware thread-specific status information is a value dependent on a logical hardware thread number of a particular

hardware thread assigned to the given partition and specified by the access, and wherein the program instructions that transform comprise program instructions for:

altering the logical hardware thread number to determine the actual hardware thread number;

retrieving the hardware thread-specific information for the actual hardware thread; and

returning the retrieved hardware thread-specific information in response to the access.

27. The computer system of Claim 26, wherein the hardware thread-specific status information is a hardware thread number of the hardware thread performing the access, and wherein the altering subtracts a base thread number for the given partition to obtain the actual hardware thread number.
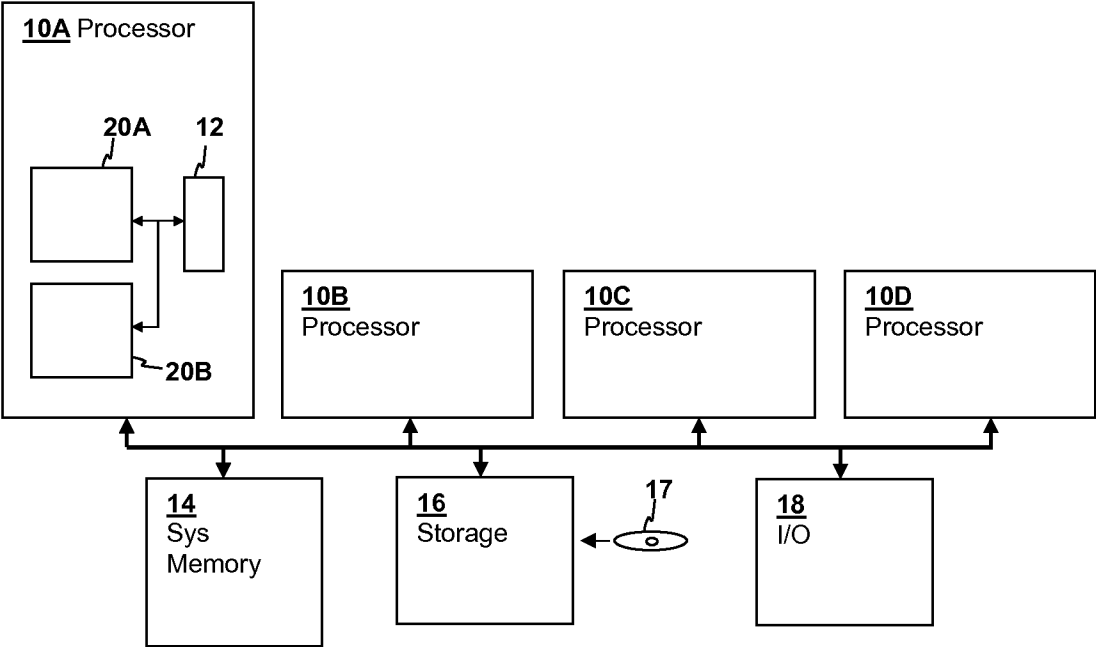
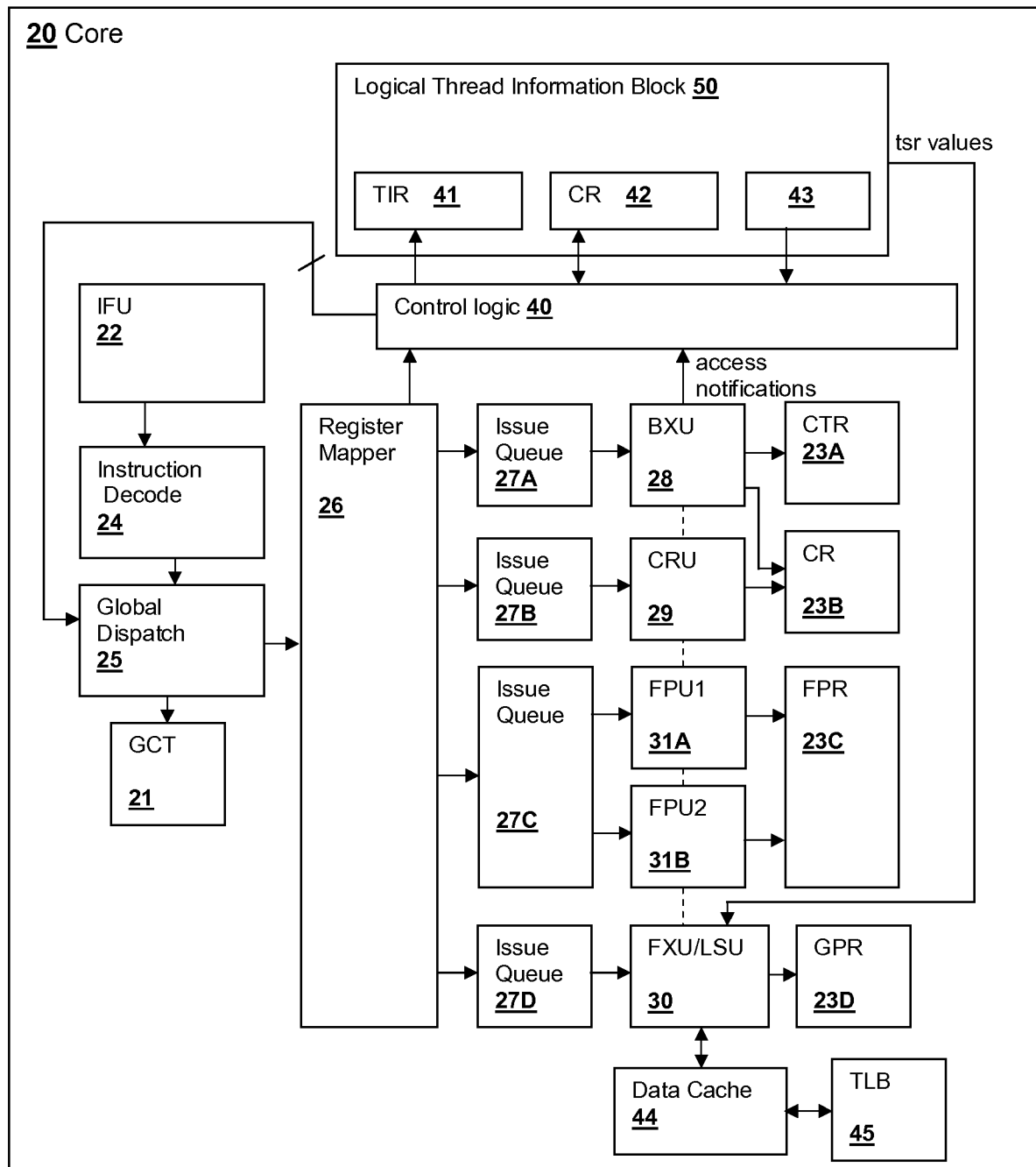**10A** Processor
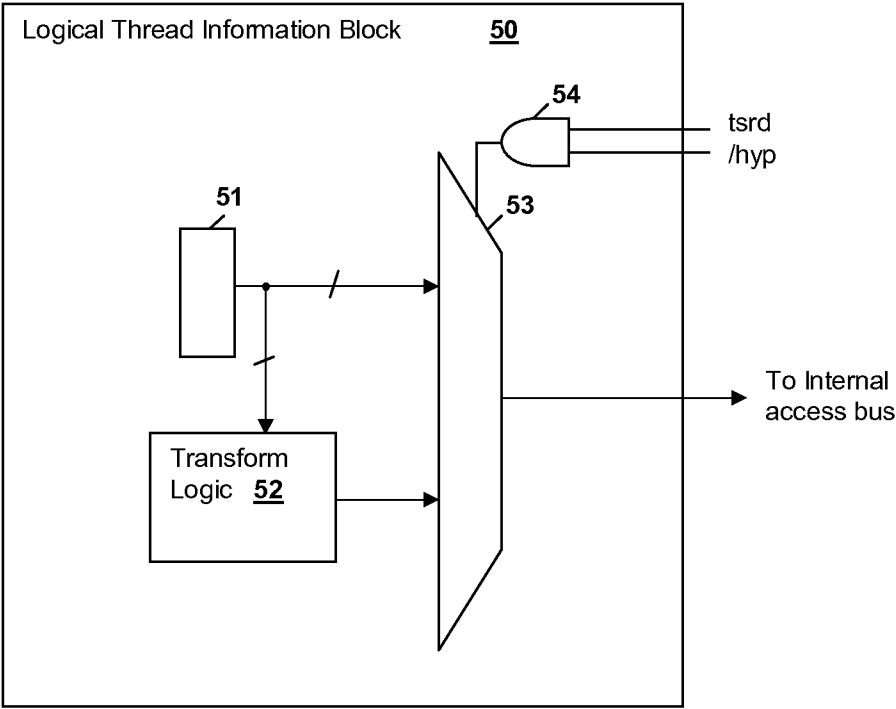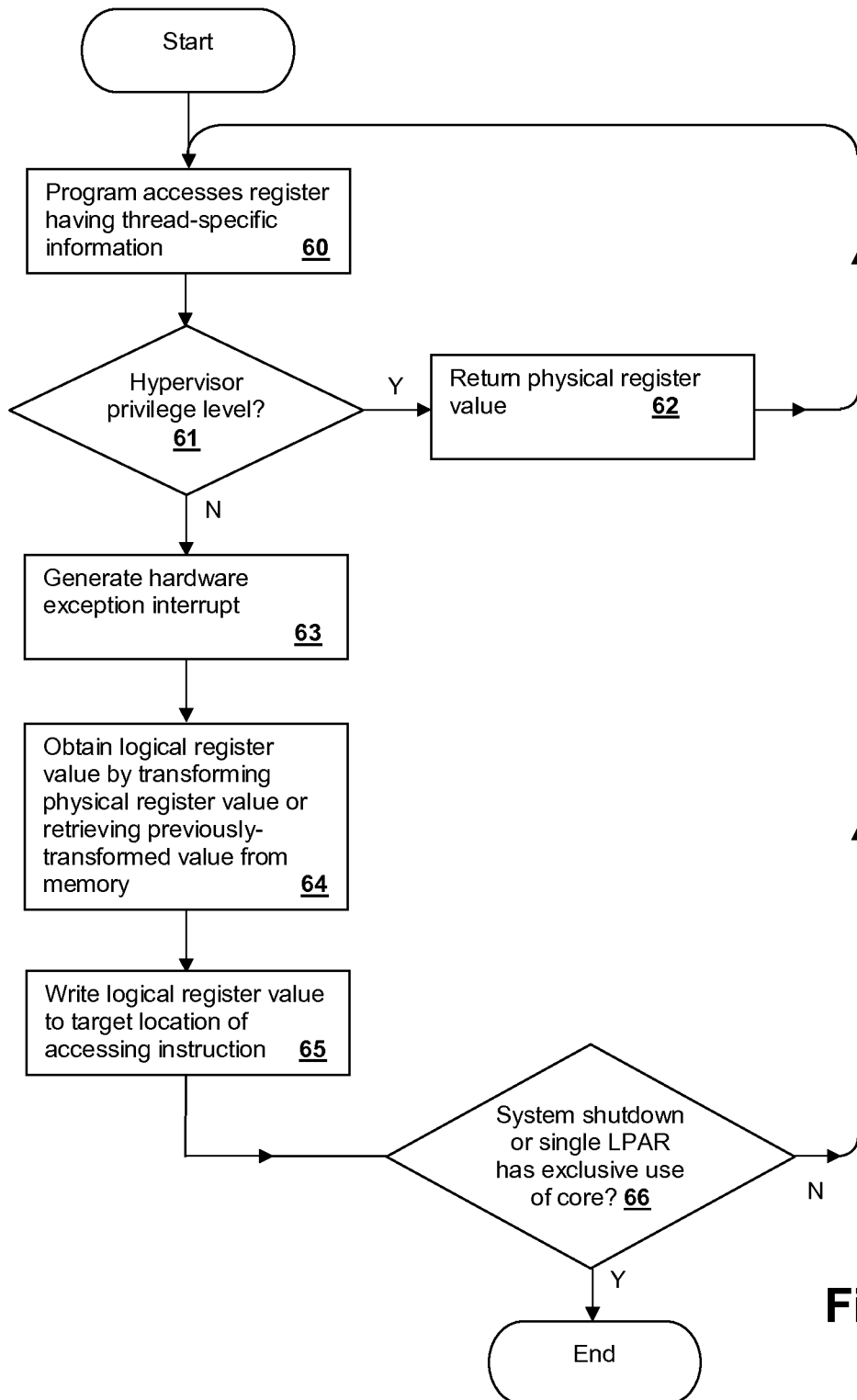
**20A**   **12**

**20B**

**10B**
Processor

**10C**
Processor

**10D**
Processor

**14**
Sys
Memory

**16**
Storage

**17**

**18**
I/O

# Fig. 1

**20** Core

Logical Thread Information Block **50**

| TIR **41** | CR **42** | **43** |

tsr values

IFU
**22**

Control logic **40**

access
notifications

Instruction Decode
**24**

Register Mapper
**26**

Issue Queue **27A**

BXU **28**

CTR **23A**

Global Dispatch
**25**

Issue Queue **27B**

CRU **29**

CR **23B**

GCT
**21**

Issue Queue
**27C**

FPU1 **31A**

FPR **23C**

FPU2 **31B**

Issue Queue **27D**

FXU/LSU **30**

GPR **23D**

Data Cache **44**

TLB **45**

## Fig. 2

**Fig. 3**

Fig. 4