METHOD FOR SCHEDULING PROCESSOR RESOURCES, INTEGRATED CIRCUIT, APPARATUS, AND ELECTRONIC DEVICE

FIELD OF THE PRESENT DISCLOSURE

[0001] The present disclosure relates to the field of automotive intelligent cockpits, and in particular, to a method for scheduling processor resources, an integrated circuit, an apparatus, and an electronic device.

BACKGROUND OF THE PRESENT DISCLOSURE

[0002] In related arts, processor cores running an operating system are fixed. In order to ensure the execution of tasks run on the operating system, the processor cores running the operating system need to be configured to satisfy a maximum demand of the operating system for processor resources.

[0003] However, if the demand of the operating system for processor resources changes, for example, if the demand of the operating system for processor resources decreases, while the processor cores running the operating system are still configured to satisfy the maximum demand of the operating system for processor resources, system resources will be wasted.

SUMMARY OF THE PRESENT DISCLOSURE

[0004] In order to solve the above technical problems, the present disclosure provides a method for scheduling processor resources, an integrated circuit, an apparatus, and an electronic device, so as to solve the problem in the prior art that allocating processors according to a maximum demand of operating systems for processor resources leads to waste of system resources.

[0005] In a first aspect of the present disclosure, there is provided a method for scheduling processor resources, the method being applied to an integrated circuit comprising a plurality of processor cores, which respectively run a plurality of operating systems, the method comprising:

[0006] obtaining demand information of the plurality of operating systems each for

processor resources; determining a first operating system and a second operating system based on the demand information of the plurality of operating systems each for processor resources, wherein the first operating system is an operating system capable of releasing processor resources, and the second operating system is an operating system requiring allocation of more processor resources; determining, based on the demand information of the first operating system for processor resources, a target processor core from processor cores on which the first operating system is run, and causing the first operating system to stop running on the target processor core; and running the second operating system on the target processor core.

[0007] In a second aspect of the present disclosure, there is provided an integrated circuit, comprising a controller and a plurality of processor cores running a plurality of operating systems, wherein the controller is electrically connected to the plurality of processor cores. The controller is configured to:

[0008] obtain demand information of the plurality of operating systems each for processor resources; determine a first operating system and a second operating system based on the demand information of the plurality of operating systems each for processor resources, wherein the first operating system is an operating system capable of releasing processor resources, and the second operating system is an operating system requiring allocation of more processor resources; determine, based on the demand information of the first operating system for processor resources, a target processor core from processor cores on which the first operating system is run; control the first operating system to stop running on the target processor core; and control the second operating system to run on the target processor core.

[0009] In a third aspect of the present disclosure, there is provided an apparatus for scheduling processor resources, applied to an integrated circuit, comprising a plurality of processor cores which respectively run a plurality of operating systems, the apparatus comprises:

[0010] an obtaining module, configured to obtain demand information of the plurality of operating systems each for processor resources; a first determination module, configured to determine a first operating system and a second operating system based

on the demand information of the plurality of operating systems each for processor resources, wherein the first operating system is an operating system capable of releasing processor resources, and the second operating system is an operating system requiring allocation of more processor resources; a second determination module, configured to determine, based on the demand information of the first operating system for processor resources, a target processor core from processor cores on which the first operating system is run, and cause the first operating system to stop running on the target processor core; and a running module, configured to run the second operating system on the target processor core.

[0011] In a fourth aspect of the present disclosure, there is provided a computer-readable storage medium, storing a computer program thereon, which, when executed by a processor, causes the processor to execute the method for scheduling processor resources proposed in the embodiments of the first aspect of the present disclosure.

[0012] In a fifth aspect of the present disclosure, there is provided an electronic device, comprising:

[0013] a processor; and

[0014] a memory for storing instructions executable by the processor;

[0015] wherein the processor is configured to read the executable instructions from the memory and execute the instructions to implement the method for scheduling processor resources proposed in the embodiments of the first aspect of the present disclosure.

[0016] The embodiments of the present disclosure provide a method for scheduling processor resources, an integrated circuit, an apparatus, and an electronic device. By obtaining demand information of a plurality of operating systems each for processor resources, and based on the demand information, determining a first operating system releasing processor resources and a second operating system requiring allocation of more processor resources, and then, determining, based on the demand information of the first operating system for processor resources, a target processor core from processor cores on which the first operating system is run, and causing the first operating system to stop running on the target processor core, so as to run the second operating system on the target processor core. The present disclosure can dynamically adjust the

number of processor cores available to respective operating systems based on the actual demands of a plurality of operating systems, so that processor resources may be allocated on demand to different operating systems without always allocating processor cores according to the maximum resource demand of each operating system, thereby avoiding idle waste of processor resources and improving the resource utilization rate of the plurality of operating systems of the entire integrated circuit.

BRIEF DESCRIPTION OF THE DRAWINGS

[0017] FIG. 1 is an architecture diagram of a system for scheduling processor resources provided by an exemplary embodiment of the present disclosure;

[0018] FIG. 2 is a schematic flowchart of a method for scheduling processor resources provided by an exemplary embodiment of the present disclosure;

[0019] FIG. 3 is one schematic flowchart of step S201 provided by an exemplary embodiment of the present disclosure;

[0020] FIG. 4 is one schematic flowchart of step S202 provided by an exemplary embodiment of the present disclosure;

[0021] FIG. 5 is one schematic flowchart of step S203 provided by an exemplary embodiment of the present disclosure;

[0022] FIG. 6 is one schematic flowchart of step S204 provided by an exemplary embodiment of the present disclosure;

[0023] FIG. 7 is one schematic flowchart of step S2041 provided by an exemplary embodiment of the present disclosure;

[0024] FIG. 8 is another schematic flowchart of step S201 provided by an exemplary embodiment of the present disclosure;

[0025] FIG. 9 is another schematic flowchart of step S2041 provided by an exemplary embodiment of the present disclosure;

[0026] FIG. 10 is a schematic structural diagram of an integrated circuit provided by an exemplary embodiment of the present disclosure;

[0027] FIG. 11 is a structural diagram of an apparatus for scheduling processor resources provided by an exemplary embodiment of the present disclosure;

[0028] FIG. 12 is a structural diagram of an electronic device provided by an embodiment of the present disclosure.

DESCRIPTION OF THE EMBODIMENTS

[0029] In order to explain the present disclosure, exemplary embodiments of the present disclosure will be described in detail below with reference to the accompanying drawings. Obviously, the described embodiments are only a part of the embodiments of the present disclosure, rather than all of the embodiments. It should be understood that the present disclosure is not limited by the exemplary embodiments.

[0030] It should be noted that, the relative arrangement of components and steps, numerical expressions, and numerical values set forth in these embodiments do not limit the scope of the present disclosure, unless specifically stated otherwise.

[0031] Application Overview

[0032] With the rapid development of automotive intelligent cockpit and cockpit-driving integration technology, in-vehicle integrated circuits need to simultaneously carry a plurality of processor cores (CPU, Central Processing Unit) and run a plurality of operating systems to collaboratively support the stable operation of diversified in-vehicle functions such as instrument display, autonomous driving, intelligent cockpit, and in-vehicle entertainment. The allocation and scheduling efficiency of processor cores, as the core computing resources of in-vehicle integrated circuits, directly determine the task execution efficiency of operating systems, the system response speed, and the overall intelligent interaction experience of the vehicle.

[0033] In a multi-operating-system scenario, two schemes are mainly adopted in processor resource allocation: one is static binding allocation of physical processor cores, that is, the in-vehicle integrated circuit allocates a fixed number of processor cores to operating systems according to the maximum demand of each operating system for processor resources. The other is virtual processor core (vCPU)scheduling based on a hypervisor, that is, the in-vehicle integrated circuit abstracts physical processor cores into virtual processor cores through the hypervisor and allocates the virtual processor cores to different operating systems, and dynamically adjusts the virtual processor core

allocation according to the load conditions of operating systems.

[0034] For the static binding allocation approach, once the in-vehicle integrated circuit allocates processor resources to a corresponding operating system, dynamic adjustment between different operating systems is not possible. When the current load of the operating system is low and the operating system does not need all the allocated processor cores to use, the excess processor cores will be in an idle state and cannot be utilized by other operating systems with higher loads that need more processor resources, thereby causing waste of processor resources and lowering the overall resource utilization rate of the system.

[0035] For the hypervisor virtualization scheduling approach, although processor resources can be dynamically allocated among different operating systems, the virtualization scheduling occupies a certain amount of processor core computing power in the process of implementing the mapping from physical CPUs to vCPUs, and introduces a certain time delay, thereby reducing the speed of processor resource allocation.

[0036] In view of the technical problems existing in the above two schemes, the processor resource scheduling scheme proposed in the embodiments of the present disclosure can dynamically adjust, according to the real-time resource demands of a plurality of operating systems, the number of physical processor cores available to each operating system, and dynamically allocate the idle physical processor cores of operating systems with reduced demands to operating systems with increased demands for use, thereby avoiding idle waste of processor resources and improving the resource utilization rate of the plurality of operating systems of the overall integrated circuit.

[0037] Exemplary System

[0038] FIG. 1 is an architecture diagram of a system for scheduling processor resources provided by an exemplary embodiment of the present disclosure. The system for scheduling processor resources may run on an integrated circuit 100. The integrated circuit 100 in the present disclosure may be an in-vehicle integrated circuit. The integrated circuit 100 may comprise a plurality of hardware units, and the plurality of hardware units comprise a plurality of processor cores 120. The integrated circuit 100

may run a plurality of operating systems 110 through the plurality of hardware units. Before running the operating systems 110, the integrated circuit 100 may allocate different numbers of processor cores 120 to different operating systems 110 according to the demand of respective operating systems 110 for processor resources.

[0039] Exemplarily, as shown in FIG. 1, two operating systems 110 are allocated with two processor cores 120 and three processor cores 120, respectively. When a demand of the operating system 110 having three processor cores 120 for processor resources is less than three processor cores 120, the integrated circuit 100 may migrate one processor core (the dashed-line processor core as shown in FIG. 1) among the three processor cores 120 of the operating system 110 as a target processor core 121 to another operating system 110 for use. For example, when the demand of the operating system currently having two processor cores 120 for processor resources increases, the target processor core 121 is migrated to the operating system 110. In this way, via the migration of the processor cores 120, the operating system 110 originally having three processor cores 120 becomes an operating system 110 having two processor cores 120, and the operating system 110 can normally run based on these two processor cores 120; the operating system 110 originally having two processor cores 120 becomes an operating system 110 having three processor cores 120 (the two original processor cores 120 and the migrated target processor core 121), which may improve the operating efficiency of the operating system 110 and fully utilize the processor resources of the integrated circuit 100.

[0040] The integrated circuit 100 comprises a firewall module 130, configured to interrupt a mapping relationship between the target processor core 121 and the original operating system 110, and to change a memory used by the target processor core 121. The firewall module 130 comprises an interrupt firewall 131 and a memory firewall 132. The interrupt firewall 131 may block the original operating system 110 from sending interrupt signals to the target processor core 121, preventing the original operating system 110 from continuing to schedule the target processor core 121 to execute tasks; the memory firewall 132 may isolate a memory space between the target processor core 121 and the original operating system 110, preventing the target processor core 121 from

continuing to access memory data of the original operating system 110, thereby ensuring the data security of the original operating system 110, and at the same time allocate a memory space corresponding to the current operating system 110 for the target processor core 121, ensuring that the current operating system 110 may normally control the target processor core 121 to complete operation.

[0041] In some examples, the integrated circuit 100 may further comprise a system control processor. The system control processor may be communicatively connected to different operating systems 110, and manage the processor cores 120 and the firewall module 130. Exemplarily, after detecting changes in the demand of different operating systems 110 for processor resources, the system control processor generates a migration instruction for the processor cores 120, sends a mapping relationship adjustment instruction to the firewall module 130, and coordinates the target processor core 121 to complete switching of an operating environment, thereby ensuring the orderly progress of the migration process, avoiding operational anomalies of different operating systems 110 during the migration process, and ensuring that the dynamic adjustment process of processor resources of the entire integrated circuit 100 is stable and reliable.

[0042] Exemplary Method

[0043] FIG. 2 is a schematic flowchart of a method for scheduling processor resources provided by an exemplary embodiment of the present disclosure. The present embodiment may be applied to an electronic device. The electronic device may comprise an integrated circuit. The integrated circuit may comprise a plurality of processor cores. The plurality of processor cores are configured to run a plurality of operating systems. As shown in FIG. 2, the above method comprises the following steps:

[0044] S201: Obtaining demand information of a plurality of operating systems each for processor resources.

[0045] The demand information of an operating system for processor resources is generated through statistics by the operating system during operation based on a load condition of tasks currently to be executed or tasks being executed, which reflects an actual demand of the operating system for computing power at a current stage.

[0046] The integrated circuit may periodically collect the demand information of each operating system for processor resources through a system control processor, or may receive the demand information for processor resources actively uploaded by the operating system when the operating system runs corresponding system tasks.

[0047] In some examples, since different operating systems correspond to different functions, the frequency and magnitude of changes in the demand of different operating systems for processor resources also differ. The integrated circuit may set differentiated collection periods for different operating systems: shortening a collection interval for an operating system with frequent changes in the demand for processor resources to improve the sensitivity of processor resource demand perception, and extending a collection interval for an operating system with gentle changes in the demand for processor resources to reduce the occupation of processor resources by the collection process itself.

[0048] S202: Determining a first operating system and a second operating system based on the demand information of the plurality of operating systems each for processor resources.

[0049] In some examples, the first operating system is an operating system releasing processor resources, and the second operating system is an operating system requiring allocation of more processor resources. The integrated circuit may determine a number of processor cores currently demanded by each operating system based on the demand information of the plurality of operating systems each for processor resources.

[0050] The integrated circuit may compare the number of processor cores currently allocated to each operating system with the number of processor cores currently demanded by the operating system. If the number of processor cores currently allocated to a certain operating system is greater than the number of processor cores currently demanded by the operating system, it indicates that the operating system has idle and releasable processor cores, and the operating system may be determined as the first operating system; if the number of processor cores currently allocated to a certain operating system is less than the number of processor cores currently demanded by the operating system, it indicates that the operating system needs more processor resources

to support task operation, and the operating system may be determined as the second operating system. If the number of processor cores currently allocated to a certain operating system is equal to the number of processor cores currently demanded by the operating system, it indicates that the processor resources of the operating system are just sufficient to support the operating system in running the corresponding tasks, and the integrated circuit will not perform any migration operation on the processor cores of the operating system.

[0051] S203: Determining, based on the demand information of the first operating system for processor resources, a target processor core from processor cores on which the first operating system is run, and causing the first operating system to stop running on the target processor core.

[0052] The target processor core is the processor core among the processor cores of the first operating system that needs to be migrated to the second operating system. After the first operating system is determined, the integrated circuit may calculate a difference between the number of processor cores currently allocated to the first operating system and the number of processor cores currently demanded by the first operating system based on the demand information of the first operating system for processor resources. The difference is the number of processor cores releasable by the first operating system.

[0053] The integrated circuit may determine the target processor core from the releasable processor cores of the first operating system. The target processor core may be determined based on an actual operating condition. For example, the integrated circuit may select a processor core with the lowest priority and the least current load as the target processor core, so as to avoid affecting normal task operation of the first operating system after selecting the processor core.

[0054] After the target processor core is selected or determined, the integrated circuit may migrate the target processor core from the first operating system to the second operating system by means of hot-plugging. Specifically, the integrated circuit may stop the first operating system from running on the target processor core, so that the target processor core goes offline from the first operating system.

[0055] In some examples, before the first operating system takes the target processor core offline, the integrated circuit may first have the first operating system migrate tasks being executed on the target processor core to other processor cores of the first operating system, and then interrupt an association between the target processor core and the first operating system through a firewall module of the integrated circuit, so as to stop the first operating system from running on the target processor core, thereby avoiding scheduling conflicts.

[0056] The number of first operating systems and the number of second operating systems may be one or more, respectively. When there are a plurality of first operating systems, the integrated circuit may preferentially select the first operating system with the largest number of releasable target processor cores to release resources. When there are a plurality of second operating systems, the second operating system with a greater demand for processor resources may be preferentially selected for allocation of the target processor cores.

[0057] S204: Running the second operating system on the target processor core.

[0058] After the first operating system takes the target processor core offline, the integrated circuit may switch an operating environment of the target processor core to an operating environment corresponding to the second operating system, so that the second operating system can run on the target processor core after the target processor core is migrated to the second operating system. At this time, the target processor core goes online in the second operating system, and the second operating system may schedule tasks awaiting execution to run on the target processor core, thereby achieving dynamic adjustment of the target processor resources.

[0059] It should be noted that, for a single operating system, it may dynamically manage the going online and offline of the corresponding processor cores, without managing the migration of processor cores. For the integrated circuit, after the target processor core goes offline from the first operating system, it is not idle but goes online in the second operating system, thereby assisting the second operating system in executing system tasks, so that the processor resources of the integrated circuit are fully utilized and the resource utilization rate is improved.

[0060] By obtaining demand information of a plurality of operating systems for processor resources, determining a first operating system releasing processor resources and a second operating system requiring more processor resources, determining a target processor core from processor cores of the first operating system and switching to run the second operating system, the method provided by the embodiments of the present disclosure can dynamically adjust allocation of physical processor cores based on actual resource demands of each operating system, without fixedly allocating processor cores according to the maximum resource demand of each operating system, thereby avoiding idle waste of processor resources and improving the resource utilization rate of the plurality of operating systems of the integrated circuit. At the same time, by directly scheduling physical processor cores, there is no computing power occupation and latency issue of a hypervisor, ensuring the efficiency of resource allocation.

[0061] FIG. 3 is one schematic flowchart of step S201 provided by an exemplary embodiment of the present disclosure.

[0062] Referring to FIG. 3, on the basis of the embodiment shown in FIG. 2 described above, step S201 may comprise the following steps:

[0063] S2011: Determining a utilization rate of a plurality of operating systems for the respectively allocated processor resources.

[0064] In some examples, the integrated circuit may, during the operation of the plurality of operating systems, obtain the utilization rate of the plurality of operating systems for the respectively allocated processor resources. The utilization rate is an occupancy rate of processor cores or a number of processor cores in use. Exemplarily, the occupancy rate of the processor cores refers to the percentage of computing power of the processor occupied by the operating system, which may represent an actual utilization of the allocated processor resources by the operating system. For example, if a certain operating system occupies half of the computing power of a processor core, the occupancy rate of the processor core by the operating system is 50%.

[0065] The number of processor cores in use refers to a number of processor cores on which the operating system is currently actually running tasks. The number generally does not exceed the current maximum number of processor cores of the operating

system. The number of processor cores in use can intuitively reflect the actual usage of the allocated processor cores by the operating system. For example, the number of processor cores allocated to a certain operating system is three. When the operating system runs a certain task, two processor cores are used to provide computing power resources for the task, and the remaining one processor core is in an idle state. At this time, the number of processor cores in use is two.

[0066] S2012: Determining the demand information of the plurality of operating systems each for processor resources based on the utilization rate of the plurality of operating systems for the respectively allocated processor resources.

[0067] In some examples, the demand information for processor resources comprises demand information for releasable processor resources and demand information for allocation of more processor resources. Exemplarily, if an average utilization rate of a certain operating system for the allocated processor resources within a preset time period is lower than a first preset threshold, it indicates that the processor resources allocated to the operating system are far more than the actual usage demand, and the demand information corresponding to the operating system may be determined as releasable part of the processor resources. The number of releasable processor cores is a difference between the number of allocated processor cores and the number of actually demanded processor cores calculated based on the utilization rate.

[0068] If the average utilization rate of the operating system for the allocated processor resources within the preset time period is higher than a second preset threshold, which is greater than the first preset threshold, it indicates that the processor resources currently allocated to the operating system are insufficient to support task operation, and the demand information corresponding to the operating system may be determined as requiring allocation of more processor resources. The number of processor cores to be allocated is a difference between the number of actually demanded processor cores calculated based on the utilization rate and the number of allocated processor cores.

[0069] If the utilization rate is between the first preset threshold and the second preset threshold, it indicates that the currently allocated processor resources match the demand, and no adjustment is needed.

[0070] The method provided by the embodiments of the present disclosure, by determining the utilization rate of the plurality of the operating systems for the allocated processor resources and determining resource demand information based on the utilization rate, can accurately quantify the actual occupation of the processor resources by each operating system. The method, by using the occupancy rate of the processor cores or the number of processor cores in use as a basis for judgment, may improve the accuracy of demand information acquisition, provides reliable data support for subsequent processor core scheduling of operating systems, thereby avoiding unreasonable processor resource scheduling caused by deviation in demand information and improving the effectiveness of processor resource allocation.

[0071] FIG. 4 is one schematic flowchart of step S202 provided by an exemplary embodiment of the present disclosure.

[0072] As shown in FIG. 4, on the basis of the embodiment shown in FIG. 2 described above, step S202 may comprise the following steps:

[0073] S2021: Determining a resource demand ratio between any two target operating systems among the plurality of operating systems based on the demand information of the plurality of operating systems each for processor resources.

[0074] In some examples, since migration of processor cores consumes a certain amount of computing power resources of the integrated circuit, the integrated circuit may determine two target operating systems among the plurality of operating systems based on the demand information of the plurality of operating systems each for processor resources, and calculate the resource demand ratio between the two target operating systems, so as to determine the first operating system and the second operating system between the two target operating systems based on the resource demand ratio. Exemplarily, if one of the two target operating systems needs to be allocated more processor cores and the other may release processor cores, the ratio of the number of released processor cores to the number of processor cores required to be allocated is the resource demand ratio.

[0075] S2022: Determining a magnitude relationship between the resource demand ratio and a preset ratio.

[0076] The preset ratio may be a ratio standard preset by the integrated circuit, used to determine whether the conditions for directly performing processor core migration between the two target operating systems are satisfied. Exemplarily, the preset ratio may be 1, used to determine whether the number of released processor cores is the same as the number of processor cores required to be allocated.

[0077] It should be noted that the present disclosure only uses the above embodiments for illustrative purposes and does not limit the specific numerical value of the preset ratio. In practical applications, the integrated circuit may dynamically adjust the preset ratio based on actual scenarios such as the functional safety levels, load conditions, and task priorities of multiple operating systems.

[0078] S2023: When the resource demand ratio is greater than the preset ratio, determining the first operating system and the second operating system based on the resource demand ratio.

[0079] In some examples, if the resource demand ratio of the two target operating systems is greater than the preset ratio, it indicates that the number of releasable processor cores in the two target operating systems is greater than the number of processor cores required to be allocated in the two target operating systems. Exemplarily, at this time, the resource demand ratio is greater than the preset ratio of 1, satisfying a trigger condition for direct migration.

[0080] It should be noted that when the resource demand ratio is equal to the preset ratio, the trigger condition for direct migration may also be satisfied, and the subsequent process of determining the first operating system and the second operating system is executed.

[0081] The integrated circuit may directly determine the operating system that needs to release processor cores in the two target operating systems as the first operating system, and determine the operating system that needs to be allocated more processor cores as the second operating system.

[0082] In some examples, if the resource demand ratio of the two target operating systems is less than the preset ratio, it indicates that the number of releasable processor cores in the two target operating systems is less than the number of processor cores

required to be allocated in the two target operating systems. For example, the total number of processor cores required to be allocated for the two target operating systems is two, but the number of releasable processor cores in the two target operating systems is only one, which cannot satisfy the number of processor cores required to be allocated. Therefore, even if processor core migration is performed, the demand information for processor resources cannot be satisfied. At this time, the integrated circuit may not perform processor core migration for the two target operating systems.

[0083] The method provided by the embodiments of the present disclosure, by first determining the resource demand ratio between two target operating systems and then comparing the ratio with the preset ratio to determine whether the migration condition is satisfied, can filter out migration operations that cannot match the processor resource demands in advance, thereby reducing unnecessary migration processes, lowering the computing power consumption caused by the migration process, and improving the overall utilization efficiency of processor resources.

[0084] FIG. 5 is one schematic flowchart of step S203 provided by an exemplary embodiment of the present disclosure.

[0085] As shown in FIG. 5, on the basis of the embodiment shown in FIG. 2 described above, step S203 may comprise the following steps:

[0086] S2031: Determining a magnitude relationship between the demand information of the first operating system for processor resources and a resource demand threshold.

[0087] In some examples, the resource demand threshold is used to indicate a threshold for the number of processor cores that the first operating system needs to release. If the number of releasable processor cores of the first operating system is greater than or equal to the resource demand threshold, it indicates that the first operating system is able to release processor resources satisfying the migration requirements, and the subsequent processor core migration process may continue to be executed.

[0088] If the number of releasable processor cores of the first operating system is less than the resource demand threshold, it indicates that the number of releasable processor cores of the first operating system is too small. Even if the migration is completed, the

utilization effect of processor resource reallocation is very limited, but the computing power resources required for migration will be consumed instead. The integrated circuit may at this time terminate the current processor core migration process without executing subsequent operations.

[0089] S2032: When the demand information of the first operating system for processor resources is greater than the resource demand threshold, obtaining a first priority of a first task running on the first operating system and a second priority of a second task running on the second operating system.

[0090] In some examples, when the demand information of the first operating system for processor resources is greater than the resource demand threshold, it indicates that the number of releasable processor cores of the first operating system is sufficient to satisfy the processor resources needed by the second operating system. At this time, the integrated circuit may respectively obtain the first priority of the first task and the second priority of the second task. The first task is a system task currently running on the first operating system, and the second task is a system task currently running on the second operating system.

[0091] In some examples, the first priority and the second priority may be determined by the integrated circuit according to priority rules preconfigured in a task management system, or may be dynamically adjusted based on a real-time running state of the tasks. A higher priority indicates that the demand of the task for processor resources is more urgent and the impact of operation interruption is greater.

[0092] S2033: If the first priority is lower than the second priority, determining a preset number of target processor cores from the processor cores on which the first operating system is run.

[0093] In some examples, if the first priority of the first task is higher than the second priority of the second task, it indicates that the first task currently running on the first operating system is more important. At this time, migrating the target processor cores from the first operating system may affect the operational stability of the high-priority first task. At this time, the migration operation may not be performed, and sufficient processor resources may be retained to support the first operating system in normally running the

first task.

[0094] If the second priority of the second task is higher than the first priority of the first task, it indicates that migrating the processor cores can ensure the computing power supply for the more important second task. At this time, the integrated circuit may determine a preset number of target processor cores from the processor cores on which the first operating system is run. The preset number may be determined based on the demand information of the second operating system. For example, the first operating system may release five processor cores, while the second operating system needs to be allocated an additional three processor cores to support the operation of the second task. At this time, the preset number may be three. Even if the first operating system has five processor cores in an idle state, only three target processor cores need to be migrated to the second operating system. The remaining two processor cores may be migrated to other operating systems, or may be used by the first operating system itself.

[0095] In some examples, the processor core with the lowest operating load may be selected from the plurality of releasable processor cores of the first operating system as the target processor core, so as to minimize the impact of processor core migration on the currently running tasks of the first operating system. For example, the integrated circuit may monitor the real-time utilization rate of each processor core in the first operating system, and preferentially select processor cores with a utilization rate below a preset threshold (such as 20%) as the target processor cores, ensuring that the migration of the processor cores will not cause a significant decrease in the operating efficiency of the first operating system.

[0096] In some examples, after determining the target processor core, the integrated circuit needs to interrupt the first operating system to stop running on the target processor core. At this time, the target processor core may transfer the currently executing task data to other processor cores of the first operating system, and then the integrated circuit may control the first operating system to stop running on the target processor core, so as to facilitate subsequent migration of the target processor core.

[0097] The method provided by the embodiments of the present disclosure, by first determining whether the releasable resources of the first operating system reach the

resource demand threshold, and then determining whether to migrate and the number of target processor cores to be migrated in combination with task priorities, can accurately screen out suitable migration objects under the premise of ensuring the stable operation of high-priority tasks, minimizing the impact of migration on the source operating system.

[0098] FIG. 6 is one schematic flowchart of step S204 provided by an exemplary embodiment of the present disclosure.

[0099] As shown in FIG. 6, on the basis of the embodiment shown in FIG. 2 described above, step S204 may comprise the following steps:

[00100] S2041: Configuring a target access permission corresponding to the second operating system for the target processor core.

[00101] In some examples, after causing the first operating system to stop running on the target processor core, the integrated circuit may clear the preconfigured access permission corresponding to the first operating system for the target processor core, and configure a target access permission corresponding to the second operating system for the target processor core, so that the second operating system can use the target processor core to access resources corresponding to the second operating system, thereby running the second task.

[00102] The target access permission is used for the second operating system to access external devices corresponding to the second operating system through the target processor core. The external devices may include hardware devices corresponding to the operating system. Exemplarily, if the second operating system is an autonomous driving operating system, the corresponding external devices may include cameras, radars, sensors, etc. If the second operating system is an intelligent cockpit operating system, the corresponding external devices may include displays, Graphics Processing Units (GPU), etc.

[00103] S2042: Running the second operating system on the target processor core based on the target access permission.

[00104] In some examples, after the integrated circuit configures the target access permission corresponding to the second operating system for the target processor core,

the second task currently awaiting execution by the second operating system may be loaded, based on the target access permission, into a memory space of the target processor core, and a running process of the second operating system on the target processor core may be started.

[00105] In some examples, the integrated circuit will monitor the running state of the target processor core in real time, including but not limited to task execution efficiency, memory occupancy rate, and whether a communication link between the target processor core and an external device corresponding to the second operating system is unobstructed. Exemplarily, when the second operating system is an autonomous driving operating system, the integrated circuit will monitor whether the target processor core can normally receive image data from cameras and perception information from radars, and ensure that such data are processed within a preset time delay (such as 50 milliseconds) to satisfy the real-time requirements of the autonomous driving tasks.

[00106] The method provided by the embodiments of the present disclosure, by configuring the target access permission corresponding to the second operating system for the target processor core, ensures that the second operating system can normally access the required external devices on the migrated processor core, achieving precise matching of hardware resources and software tasks. At the same time, by monitoring the running state of the target processor core and the communication link with the external devices in real time, anomalies that may occur during operation can be discovered and handled in a timely manner. For example, when the task execution efficiency is lower than a preset threshold or a communication delay exceeds a specified time, the integrated circuit may trigger an early warning mechanism or dynamically adjust a resource allocation strategy, further ensuring the stable and efficient operation of the second operating system on the target processor core, and improving the resource utilization flexibility and task execution reliability of the entire system.

[00107] FIG. 7 is one schematic flowchart of step S2041 provided by an exemplary embodiment of the present disclosure.

[00108] As shown in FIG. 7, on the basis of the embodiment shown in FIG. 6 described above, step S2041 may comprise the following steps:

[00109] S20411: Obtaining a target memory address corresponding to the second operating system.

[00110] In some examples, the target memory address corresponding to the second operating system is a specific memory region used by the second operating system during operation for storing program code, data, and running state information. The target memory address may be pre-allocated by a Memory Management Unit (MMU) of the integrated circuit and recorded in a system configuration table. The integrated circuit may quickly obtain the target memory address corresponding to the second operating system by querying the system configuration table.

[00111] S20412: Determining a memory firewall configuration of the target processor core based on the target memory address.

[00112] In some examples, the integrated circuit may determine the memory firewall configuration of the target processor core based on the target memory address. The memory firewall configuration is a set of access rules preconfigured by a memory firewall for the target processor core, used to restrict an access range of the target processor core to a memory space.

[00113] Specifically, the memory firewall configuration will specify the memory regions that the target processor core is allowed to access and the memory regions that it is prohibited from accessing. Before the integrated circuit migrates the target processor core to the second operating system, the memory firewall configuration corresponding to the target processor core is used to indicate that the target processor core is allowed to access the memory spaces of the first operating system and is prohibited from accessing the memory spaces of other operating systems (including the second operating system).

[00114] S20413: Based on the memory firewall configuration, configuring a first target access permission corresponding to the second operating system for the target processor core.

[00115] In some examples, after migrating the target processor core to the second operating system, the integrated circuit needs to change the memory firewall configuration of the target processor core, so that the target processor core after the

configuration change can normally access the second operating system and is prohibited from accessing other operating systems (including the first operating system).

[00116] To this end, the integrated circuit may configure a first target access permission corresponding to the second operating system for the target processor core. The first target access permission is used to indicate that the target processor core is only capable of accessing the target memory address corresponding to the second operating system and is incapable of accessing the memory address corresponding to the first operating system, thereby achieving physical isolation of the memory spaces of different operating systems, preventing the target processor core from accessing the memory space of the original operating system (the first operating system) across operating systems due to the memory configuration not being changed in a timely manner, which would result in data leakage or system crash. For example, when running the second operating system, if the target processor core attempts to read the memory data of the first operating system, the memory firewall will immediately block the access request and feed back anomaly information to the integrated circuit, ensuring the independence and security of the memory space of each operating system.

[00117] The method provided by the embodiments of the present disclosure, by obtaining the target memory address corresponding to the second operating system, determining the memory firewall configuration of the target processor core, and then configuring the first target access permission corresponding to the second operating system for the target processor core, may achieve strict physical isolation of the memory spaces of different operating systems, effectively preventing the target processor core from accessing the memory data of the original operating system (the first operating system) after migration due to untimely configuration changes, and safeguarding the independence and data security of the memory space of each operating system at the hardware level.

[00118] FIG. 8 is another schematic flowchart of step S201 provided by an exemplary embodiment of the present disclosure.

[00119] As shown in FIG. 8, on the basis of the embodiment shown in FIG. 2 described above, step S201 may comprise the following steps:

[00120] S2013: Obtaining a running state of a vehicle.

[00121] In some examples, the running state of the vehicle is used to indicate that the vehicle is in a stopped state or the vehicle is in a driving state. For example, the integrated circuit may obtain the running state information of the vehicle through a vehicle speed sensor or a parking sensor of the vehicle, and determine whether the vehicle is currently in different states such as a stopped state or a driving state.

[00122] S2014: Determining demand information of the plurality of operating systems each for processor resources according to the running state of the vehicle.

[00123] Under different running states of the vehicle, the demand information of the plurality of operating systems each for processor resources also differs. The integrated circuit may determine the demand information of the plurality of operating systems each for processor resources according to the running state of the vehicle. Exemplarily, taking an intelligent cockpit system and an autonomous driving operating system among the plurality of operating systems as an example, when the running state of the vehicle is in a stopped state, the autonomous driving operating system does not need to continuously perform complex environment perception, path planning, and decision control, and the demand for processor core resources decreases, while the intelligent cockpit system often needs to run high-load applications such as multimedia entertainment and navigation interaction, and the demand for processor resources increases. Conversely, when the running state of the vehicle is in a driving state, the autonomous driving operating system needs to continuously perform complex environment perception, path planning, and decision control, and the demand for processor core resources increases. At this time, the intelligent cockpit system only needs to maintain basic interactive functions, and the demand for processor resources is relatively low.

[00124] The method provided by the embodiments of the present disclosure, by dynamically determining the demand of the plurality of operating systems for processor resources based on the running state of the vehicle, may flexibly adjust the allocation of processor cores based on the actual usage scenario of the vehicle, avoiding resource idleness or insufficiency caused by fixed allocation of hardware resources, and improving the rationality of processor resource allocation by the integrated circuit for the plurality of

operating systems.

[00125] FIG. 9 is another schematic flowchart of step S2041 provided by an exemplary embodiment of the present disclosure.

[00126] As shown in FIG. 9, on the basis of the embodiment shown in FIG. 6 described above, step S2041 may comprise the following steps:

[00127] S20414: Obtaining a type of an external device corresponding to the second operating system.

[00128] In some examples, after migrating the target processor core to the second operating system, the integrated circuit needs to obtain the external devices required for the operation of the second operating system, so as to determine the type of the external devices. Exemplarily, the type of external devices corresponding to the operating system may be storage devices, input/output means, communication devices, perception devices, etc.

[00129] S20415: Determining an interrupt number corresponding to the second operating system according to the type of the external device corresponding to the second operating system.

[00130] The interrupt number is an urgent signal sent by the external device to the corresponding operating system, used to instruct the corresponding operating system to prioritize the execution of a system task corresponding to the external device.

[00131] Different types of external devices correspond to different interrupt numbers. The integrated circuit may determine the interrupt number corresponding to the second operating system according to the type of the external devices currently connected to the second operating system. Exemplarily, the external devices corresponding to the autonomous driving operating system may include cameras, radars, and sensors, and the interrupt numbers of the second operating system may include an interrupt number of the camera, an interrupt number of the radar, and an interrupt number of the sensor.

[00132] S20416: Determining an interrupt request sending address corresponding to the interrupt number from a preset identification table according to the interrupt number.

[00133] In some examples, the preset identification table is used to store an association relationship between interrupt numbers and interrupt request sending addresses. In

order to distinguish interrupt numbers from different external devices, each interrupt number may correspond to one interrupt request sending address. The interrupt request sending address is a target address to which the external device needs to transmit when sending an interrupt request. Different interrupt numbers correspond to interrupt request sending addresses of different operating systems, which can avoid interrupt requests being erroneously sent to other operating systems and affecting the normal operation of other operating systems.

[00134] S20417: Determining an interrupt firewall configuration of the target processor core based on the interrupt request sending address.

[00135] In some examples, after obtaining the interrupt request sending address corresponding to the second operating system, the integrated circuit may determine the interrupt firewall configuration of the target processor core based on the interrupt request sending address. The interrupt firewall configuration is a set of interrupt access rules preconfigured by the interrupt firewall for the target processor core, used to restrict a sending range of interrupt requests from external devices, ensuring that interrupt requests are only sent to the target processor core of the corresponding operating system and are not erroneously sent to processor cores of other operating systems.

[00136] Before the target processor core is migrated to the second operating system, the target processor core works for the first operating system, and the corresponding interrupt firewall configuration allows the target processor core to receive interrupt requests from external devices associated with the first operating system, and prohibits the target processor core from receiving interrupt requests from external devices associated with the second operating system.

[00137] S20418: Based on the interrupt firewall configuration, configuring a second target access permission corresponding to the second operating system for the target processor core.

[00138] In some examples, after migrating the target processor core to the second operating system, in order to prevent the target processor core from continuing to receive interrupt numbers from the first operating system, the integrated circuit will configure the second target access permission corresponding to the second operating system for the

target processor core through the interrupt firewall. The second target access permission indicates that interrupt requests from external devices are directed to the second operating system. In this way, the target processor core may only receive interrupt numbers sent by the external devices of the second operating system, and is prohibited from receiving interrupt numbers sent by other operating systems (including the first operating system).

[00139] In some examples, after the second operating system runs the second task, the target processor core may be released back to the first operating system, and the memory access permission and the interrupt access permission between the target processor core and the first operating system may be restored through the memory firewall and the interrupt firewall.

[00140] The method provided by the embodiments of the present disclosure, by configuring the second target access permission of the second operating system for the target processor core, may ensure that the target processor core only responds to interrupt numbers from the second operating system, avoiding the target processor core from receiving interrupt numbers sent by the original operating system (the first operating system) after migration, improving the operational stability when the plurality of operating systems share processor resources, and also safeguarding the normal operation of external devices of each operating system.

[00141] Exemplary Integrated Circuit

[00142] FIG. 10 is a schematic structural diagram of an integrated circuit provided by an exemplary embodiment of the present disclosure.

[00143] As shown in FIG. 10, the integrated circuit 1000 may comprise a controller 1010 and a plurality of processor cores 1020. The plurality of processor cores 1020 run a plurality of operating systems. The controller 1010 is electrically connected to the plurality of processor cores 1020. The controller 1010 is configured to obtain demand information of the plurality of operating systems each for processor resources; determine a first operating system and a second operating system based on the demand information of the plurality of operating systems each for processor resources, wherein the first operating system is an operating system capable of releasing processor

resources, and the second operating system is an operating system requiring allocation of more processor resources; based on the demand information of the first operating system for processor resources, determine a target processor core from the processor cores 1020 on which the first operating system is run; control the first operating system to stop running on the target processor core; and control the second operating system to run on the target processor core.

[00144] In some examples, the integrated circuit 1000 further comprises a memory processing circuit 1030. The memory processing circuit 1030 comprises a memory firewall. The controller 1010 is further configured to obtain a target memory address corresponding to the second operating system, and to determine a memory firewall configuration of the target processor core based on the target memory address. The memory processing circuit 1030 is configured to, based on the memory firewall configuration, configure a first target access permission corresponding to the second operating system for the target processor core. The first target access permission indicates that the target processor core is enabled to access the target memory address corresponding to the second operating system and does not access the memory address corresponding to the first operating system.

[00145] In some examples, the integrated circuit 1000 further comprises an interrupt processing circuit 1040. The interrupt processing circuit 1040 comprises an interrupt firewall. The controller 1010 is further configured to obtain a type of an external device corresponding to the second operating system, determine an interrupt number corresponding to the second operating system according to the type of the external device corresponding to the second operating system, and determine, from a preset identification table, an interrupt request sending address corresponding to the interrupt number according to the interrupt number. The preset identification table stores an association relationship between the interrupt numbers and the interrupt request sending addresses. The interrupt processing circuit 1040 is configured to, based on the interrupt firewall configuration, configure a second target access permission corresponding to the second operating system for the target processor core. The second target access permission indicates that interrupt requests from external devices are directed to the

second operating system.

[00146] The beneficial technical effects corresponding to the present exemplary embodiment of the integrated circuit 1000 may refer to the corresponding beneficial technical effects in the above Exemplary Method section, and will not be repeated herein.

[00147] Exemplary Apparatus

[00148] FIG. 11 is a structural diagram of an apparatus for scheduling processor resources provided by an exemplary embodiment of the present disclosure. The apparatus is applied to an integrated circuit. The integrated circuit comprises a plurality of processor cores. The plurality of processor cores run a plurality of operating systems. The apparatus for scheduling processor resources may be used to implement the corresponding method embodiments of the present disclosure.

[00149] As shown in FIG. 11, the apparatus 1100 for scheduling processor resources comprises an obtaining module 1110, a first determination module 1120, a second determination module 1130, and a running module 1140.

[00150] The obtaining module 1110 is configured to obtain demand information of the plurality of operating systems each for processor resources; the first determination module 1120 is configured to determine a first operating system and a second operating system based on the demand information of the plurality of operating systems each for processor resources, wherein the first operating system is an operating system releasing processor resources, and the second operating system is an operating system requiring allocation of more processor resources; the second determination module 1130 is configured to, based on the demand information of the first operating system for processor resources, determine a target processor core from the processor cores on which the first operating system is run, and cause the first operating system to stop running on the target processor core; the running module 1140 is configured to run the second operating system on the target processor core.

[00151] In some examples, the obtaining module 1110 is configured to determine the utilization rate of the plurality of operating systems for the respectively allocated processor resources, and the utilization rate is an occupancy rate of processor cores or a number of processor cores in use; and to determine the demand information of the

plurality of operating systems for processor resources based on the utilization rate of the plurality of operating systems for the respectively allocated processor resources.

[00152] In some examples, the first determination module 1120 is configured to, based on the demand information of the plurality of operating systems each for processor resources, determine a resource demand ratio between any two target operating systems among the plurality of operating systems; determine a magnitude relationship between the resource demand ratio and a preset ratio; and when the resource demand ratio is greater than the preset ratio, determine the first operating system and the second operating system based on the resource demand ratio.

[00153] In some examples, the second determination module 1130 is configured to determine a magnitude relationship between the demand information of the first operating system for processor resources and a resource demand threshold; when the demand information of the first operating system for processor resources is greater than the resource demand threshold, obtain a first priority of a first task running on the first operating system and a second priority of a second task running on the second operating system; and if the first priority is lower than the second priority, determine a preset number of target processor cores from the processor cores on which the first operating system is run.

[00154] In some examples, the running module 1140 is configured to configure a target access permission corresponding to the second operating system for the target processor core, and the target access permission is used by the second operating system to access external devices corresponding to the second operating system through the target processor core; and run the second operating system on the target processor core based on the target access permission.

[00155] In some examples, the running module 1140 is configured to obtain a target memory address corresponding to the second operating system; determine a memory firewall configuration of the target processor core based on the target memory address; and based on the memory firewall configuration, configure a first target access permission corresponding to the second operating system for the target processor core. The first target access permission indicates that the target processor core is enabled to

access the target memory address corresponding to the second operating system and does not access the memory address corresponding to the first operating system.

[00156] In some examples, the obtaining module 1110 is configured to obtain a running state of a vehicle, and the running state is used to indicate that the vehicle is in a stopped state or the vehicle is in a driving state; and determine demand information of the plurality of operating systems each for processor resources according to the running state of the vehicle.

[00157] In some examples, the running module 1140 is configured to obtain a type of an external device corresponding to the second operating system; determine an interrupt number corresponding to the second operating system according to the type of the external device corresponding to the second operating system; determine, from a preset identification table, an interrupt request sending address corresponding to the interrupt number according to the interrupt number, and the preset identification table stores an association relationship between interrupt numbers and interrupt request sending addresses; determine an interrupt firewall configuration of the target processor core based on the interrupt request sending address; and based on the interrupt firewall configuration, configure a second target access permission corresponding to the second operating system for the target processor core. The second target access permission indicates that interrupt requests from external devices are directed to the second operating system.

[00158] The beneficial technical effects corresponding to the present exemplary embodiment of the apparatus may refer to the corresponding beneficial technical effects in the above Exemplary Method section, and will not be repeated herein.

[00159] Exemplary Electronic Device

[00160] FIG. 12 is a structural diagram of an electronic device provided by an embodiment of the present disclosure. As shown in FIG. 12, the electronic device 1200 comprises at least one processor 1201 and a memory 1202.

[00161] The processor 1201 may be a Central Processing Unit (CPU) or other forms of processing units having data processing capabilities and/or instruction execution capabilities, and may control other components in the electronic device 1200 to perform

desired functions.

[00162] The memory 1202 may comprise one or more computer program products. The computer program products may include various forms of computer-readable storage media, such as volatile memory and/or non-volatile memory. The volatile memory may include, for example, Random Access Memory (RAM) and/or cache memory. The non-volatile memory may include, for example, Read-Only Memory (ROM), hard disk, flash memory, etc. One or more computer program instructions may be stored on the computer-readable storage media. The processor 1201 may be configured to run one or more computer program instructions to implement the method for scheduling processor resources in the various embodiments of the present disclosure described above and/or other desired functions.

[00163] In one example, the electronic device 1200 may further comprise: an input means 1203 and an output means 1204, which are interconnected through a bus system and/or other forms of connection mechanisms (not shown).

[00164] The input means 1203 may further include, for example, keyboards, mice, etc.

[00165] The output means 1204 may output various information externally. The output means 1204 may include, for example, displays, speakers, printers, and communication networks and remote output devices connected thereto, etc.

[00166] For simplicity, only some of the components in the electronic device 1200 related to the present disclosure are shown in FIG. 12, and components such as buses, input/output interfaces, etc. are omitted. In addition, depending on specific application, the electronic device 1200 may further include any other appropriate components.

[00167] Exemplary Computer Program Product and Computer-Readable Storage Medium

[00168] In addition to the above methods and devices, embodiments of the present disclosure may further provide one computer program product, comprising computer program instructions, which, when run by a processor, cause the processor to execute the steps in the method for scheduling processor resources in the various embodiments of the present disclosure described in the “Exemplary Method” section above.

[00169] The computer program product may be written in any combination of one or

more programming languages for executing program code for the operations of the embodiments of the present disclosure. The programming languages include object-oriented programming languages such as Java, C++, etc., as well as conventional procedural programming languages such as the “C” language or similar programming languages. The program code may be executed entirely on a user computing device, partially on the user device, as a standalone software package, partially on the user computing device and partially on a remote computing device, or entirely on a remote computing device or server.

[00170] In addition, the embodiments of the present disclosure may also be a computer-readable storage medium having computer program instructions stored thereon, which, when run by a processor, cause the processor to execute the steps in the method for scheduling processor resources in the various embodiments of the present disclosure described in the “Exemplary Method” section above.

[00171] The computer-readable storage medium may adopt any combination of one or more readable media. The readable medium may be a readable signal medium or a readable storage medium. The readable storage medium may include, for example but not limited to, electrical, magnetic, optical, electromagnetic, infrared, or semiconductor systems, apparatus, or devices, or any combination of the above. More specific examples (a non-exhaustive list) of the readable storage media include: electrical connections having one or more wires, portable disks, hard disks, Random Access Memory (RAM), Read-Only Memory (ROM), Erasable Programmable Read-Only Memory (EPROM or flash memory), optical fibers, portable Compact Disc Read-Only Memory (CD-ROM), optical storage devices, magnetic storage devices, or any suitable combination of the above.

[00172] The basic principles of the present disclosure have been described above in conjunction with specific embodiments. However, the advantages, benefits, effects, etc. mentioned in the present disclosure are merely examples and not limitations, and should not be considered as necessarily possessed by each of the embodiments of the present disclosure. In addition, the specific details disclosed above are only for the purposes of illustration and ease of understanding, and not for limitation. The above details do not

limit the present disclosure to necessarily adopting the above specific details for implementation.

[00173] Those skilled in the art may make various modifications and variations to the present disclosure without departing from the spirit and scope of the present application. Thus, if such modifications and variations of the present application fall within the scope of the claims of the present disclosure and the equivalents, the present disclosure is also intended to encompass such modifications and variations.

CLAIMS:

1. A method for scheduling processor resources, the method being applied to an integrated circuit comprising a plurality of processor cores, which respectively run a plurality of operating systems, the method comprising:

obtaining demand information of the plurality of operating systems each for processor resources;

determining a first operating system and a second operating system based on the demand information of the plurality of operating systems each for processor resources, wherein the first operating system is an operating system capable of releasing processor resources, and the second operating system is an operating system requiring allocation of more processor resources;

determining, based on the demand information of the first operating system for processor resources, a target processor core from the processor cores on which the first operating system is run, and causing the first operating system to stop running on the target processor core; and

running the second operating system on the target processor core.

2. The method according to claim 1, wherein the obtaining demand information of the plurality of operating systems each for processor resources comprises:

determining a utilization rate of the plurality of operating systems for the respectively allocated processor resources; wherein the utilization rate is an occupancy rate of processor cores or a number of processor cores in use; and

determining, based on the utilization rate of the plurality of operating systems for the respectively allocated processor resources, the demand information of the plurality of operating systems each for processor resources.

3. The method according to claim 2, wherein the determining a first operating system and a second operating system based on the demand information of the plurality of operating systems each for processor resources comprises:

determining a resource demand ratio between any two target operating systems among the plurality of operating systems based on the demand information of the plurality of operating systems each for processor resources;

determining a magnitude relationship between the resource demand ratio and a

preset ratio; and

determining the first operating system and the second operating system based on the resource demand ratio when the resource demand ratio is greater than the preset ratio.

4. The method according to claim 1, wherein the determining, based on the demand information of the first operating system for processor resources, a target processor core from the processor cores on which the first operating system is run comprises:

determining a magnitude relationship between the demand information of the first operating system for processor resources and a resource demand threshold;

obtaining a first priority of a first task running on the first operating system and a second priority of a second task running on the second operating system, when the demand information of the first operating system for processor resources is greater than the resource demand threshold; and

determining a preset number of the target processor cores from the processor cores on which the first operating system is run if the first priority is lower than the second priority.

5. The method according to claim 1, wherein the running the second operating system on the target processor core comprises:

configuring a target access permission corresponding to the second operating system for the target processor core, wherein the target access permission is used by the second operating system to access external devices corresponding to the second operating system through the target processor core; and

running the second operating system on the target processor core based on the target access permission.

6. The method according to claim 5, wherein the configuring the target access permission corresponding to the second operating system for the target processor core comprises:

obtaining a target memory address corresponding to the second operating system;

determining a memory firewall configuration of the target processor core based on the target memory address; and

configuring, based on the memory firewall configuration, a first target access

permission corresponding to the second operating system for the target processor core, wherein the first target access permission indicates that the target processor core is capable of accessing the target memory address corresponding to the second operating system and is incapable of accessing the memory address corresponding to the first operating system.

7. The method according to claim 1, wherein the obtaining demand information of the plurality of operating systems each for processor resources comprises:

obtaining a running state of a vehicle, wherein the running state is used to indicate that the vehicle is in a stopped state or the vehicle is in a driving state; and

determining the demand information of the plurality of operating systems each for processor resources according to the running state of the vehicle.

8. The method according to claim 5, wherein the configuring the target access permission corresponding to the second operating system for the target processor core comprises:

obtaining a type of an external device corresponding to the second operating system;

determining an interrupt number corresponding to the second operating system according to the type of the external device corresponding to the second operating system;

determining, from a preset identification table, an interrupt request sending address corresponding to the interrupt number according to the interrupt number, wherein the preset identification table stores an association relationship between interrupt numbers and interrupt request sending addresses;

determining an interrupt firewall configuration of the target processor core based on the interrupt request sending address; and

configuring, based on the interrupt firewall configuration, a second target access permission corresponding to the second operating system for the target processor core, wherein the second target access permission indicates that interrupt requests from external devices are directed to the second operating system.

9. An integrated circuit, comprising a controller and a plurality of processor cores running a plurality of operating systems, wherein the controller is electrically connected to the plurality of processor cores; and the controller is configured to:

obtain demand information of the plurality of operating systems each for processor resources;

determine a first operating system and a second operating system based on the demand information of the plurality of operating systems each for processor resources, wherein the first operating system is an operating system capable of releasing processor resources, and the second operating system is an operating system requiring allocation of more processor resources;

determine, based on the demand information of the first operating system for processor resources, a target processor core from the processor cores on which the first operating system is run;

control the first operating system to stop running on the target processor core; and

control the second operating system to run on the target processor core.

10. The integrated circuit according to claim 9, further comprising:

a memory processing circuit;

wherein the controller is further configured to obtain a target memory address corresponding to the second operating system, and to determine a memory firewall configuration of the target processor core based on the target memory address; and

wherein the memory processing circuit is configured to configure, based on the memory firewall configuration, a first target access permission corresponding to the second operating system for the target processor core, wherein the first target access permission indicates that the target processor core is enabled to access the target memory address corresponding to the second operating system without accessing the memory address corresponding to the first operating system.

11. The integrated circuit according to claim 9, further comprising: an interrupt processing circuit;

wherein the controller is further configured to obtain a type of an external device corresponding to the second operating system, determine an interrupt number corresponding to the second operating system according to the type of the external device corresponding to the second operating system, and determine, from a preset identification table, an interrupt request sending address corresponding to the interrupt number according to the interrupt number, wherein the preset identification table stores

an association relationship between interrupt numbers and interrupt request sending addresses; and

wherein the interrupt processing circuit is configured to configure, based on an interrupt firewall configuration, a second target access permission corresponding to the second operating system for the target processor core, wherein the second target access permission indicates that interrupt requests from external devices are directed to the second operating system.

12. An apparatus for scheduling processor resources, applied to an integrated circuit comprising a plurality of processor cores, which respectively run a plurality of operating systems, the apparatus comprising:

an obtaining module, configured to obtain demand information of the plurality of operating systems each for processor resources;

a first determination module, configured to determine a first operating system and a second operating system based on the demand information of the plurality of operating systems each for processor resources, wherein the first operating system is an operating system capable of releasing processor resources, and the second operating system is an operating system requiring allocation of more processor resources;

a second determination module, configured to determine, based on the demand information of the first operating system for processor resources, a target processor core from the processor cores on which the first operating system is run, and cause the first operating system to stop running on the target processor core; and

a running module, configured to run the second operating system on the target processor core.

13. The apparatus for scheduling processor resources according to claim 12, wherein the obtaining module is configured to determine utilization rates of the plurality of operating systems for the respectively allocated processor resources, and the utilization rates are occupancy rates of processor cores or a number of processor cores in use; and to determine the demand information of the plurality of operating systems for processor resources based on the utilization rates of the plurality of operating systems for the respectively allocated processor resources.

14. A computer-readable storage medium, storing a computer program thereon,

which, when executed by a processor, causes the processor to execute the method for scheduling processor resources according to any one of claims 1 to 8.

15. An electronic device, comprising:

a processor; and

a memory for storing instructions executable by the processor;

wherein the processor is configured to read the executable instructions from the memory and execute the instructions to implement the method for scheduling processor resources according to any one of claims 1 to 8.