Hypervisor: Requirement Document (Version 3)

Jean-Raymond Abrial and Rustan Leino

No Institute Given

1 Requirement Document

1.1 A single system memory handling

- SM-0: An operating system (OS) makes use of some number of cores (CPUs), uses some memory, and communicates with some number of devices and timers.

Note: The devices may include a disk that the OS uses to implement a virtual memory. Though the state of such disks and other devices are part of the state of the OS, it is not part of the state of the memory of the OS.

- SM-1: The amount of memory accessible to an OS is determined at boot time, and is addressed as pages from 0 onwards. The addresses of these pages are called intermediate physical addresses (IPAs).

- SM-2: An OS has access to a number of per-core hardware registers, including the Translation Look-aside Buffer (MMU-TLB).

- SM-3: The MMU-TLB is an associative memory. It contains pairs of the following form: "Virtual Address - IPA"

Note: There are four ways that an OS can produce an IPA:

(0) Running in kernel mode on a core, the OS can issue instructions that address an IPA directly.

(1) Running in user mode on a core, the OS can produce a virtual address, which is translated via the MMU-TLB into an IPA.

(2) Running in user mode on a core, the OS can produce a virtual address which is not translated via the MMU-TLB (because the pair virtual address -IPA is not present in the MMU-TLB), however the IPA is produced by walking on the page table of the OS, the MMU-TLB is updated by entering the new pair and evicting an old one that is present in the MMU-TLB.

(3) Running in user mode on a core, the OS can produce a virtual address which is not translated via the MMU-TLB (because the pair virtual-address - IPA is not present in the MMU-TLB), the IPA is not produced either by walking on the page table of the OS because the corresponding virtual page is not in the memory, however the page is in the disk memory from which it can be loaded in the central memory by evicting a page in the memory, the page table is then updated and the MMU-TLB as well. The behavior described in this case, is called a first-level page fault.

- SM-5: If the OS tries to access its memory through a non-existing IPA, then it crashes.

1.2 A single system Interrupt handling

Note: The following requirements are not independent (there are some redundancies): this is done on purpose.

- SI-0: We consider uni-processor systems.

- SI-1: An OS can be interrupted by some devices that are connected to the CPU running the OS.

- SI-2: Each interrupt has a priority attached to it. The priority is a natural number.

- SI-3: An interrupt can be in one of the following state: inactive, pending, or active. The sate of an interrupt is managed by the hardware.

- SI-4: An inactive interrupt is one that has not happened since it was last treated.

- SI-5: A pending interrupt is one that has happened but is not treated yet because its priority is not greater than those of the active interrupts. When an interrupt occurs, it is made pending by the hardware.

- SI-6: An active interrupt is one that has happened and is either treated by the CPU or preempted by an interrupt of higher priority. Only active interrupts can be executed by the OS.

- SI-7: A pending interrupt with a high enough priority results in the hardware sending a signal to the concerned core of the OS. The OS acknowledges this signal and then the hardware makes the interrupt active.

- SI-8: Among the active interrupts, the one that is executed by the corresponding Interrupt Service Routine (ISR) is the one with the highest priority.

- SI-9: A running interrupt can be preempted by a new pending interrupt with a priority that is higher than that of the running interrupt. The preempted interrupt remains active although it is not running any more. It will be resumed once the preempting interrupt will be treated.

- SI-10: Once the ISR has treated an active interrupt, it sends an End Of Interrupt (EOI) signal to the hardware, which then makes the interrupt inactive.

- SI-11: The OS can mask (and unmask) some interrupts. This concerns the pending or inactive interrupts. An interrupt that is masked becomes inactive if pending. A masked interrupt cannot become pending or active until it is unmasked.

1.3 Multiple systems

- MS-0: From an abstract point of view, we imagine we have several OSs sitting next to each other, executing independently.

- MS-1: Each such OS is as defined above for a single system.

1.4 Hypervisor

- HV-0: The role of the hypervisor is to simulate on a single machine (possibly with several cores) the behavior of independent operating systems, here known as guest OSs.

- HV-1: The number of guests is fixed and is determined at boot time.

- HV-2: A guest OS is not aware that it is being executed under the hypervisor.

- HV-3: Guests are really independent: A guest cannot inspect or influence the behavior of other guests, not even know of their presence. Except: The guest may detect different performance characteristics than on a non-hypervisor machine.

- HV-4: It is the responsibility of the hypervisor to schedule the guests (see section 1.6).

- HV-5: The hypervisor itself should not take the control of its hardware permanently. That is, it should do some scheduling of guests.

1.5 Hypervisor memory handling

- HM-0: The hypervisor controls a physical memory, into which it embeds the memory of its guests.

- HM-1: The hypervisor has a data structure called the SLAT (second-level address translation), which keeps track of, for each guest, a one-to-one mapping between the guest's IPAs and the physical addresses (PAs).

- HM-2: The size and contents of the SLAT associated with each guest is determined at boot time.

- HM-3: The SLAT of each guest is made available through a base address (BA).

- HM-4: When a guest provides an IPA, the hardware attempts to map that IPA to a PA by consulting the SLAT associated with this guest.

- HM-5: If the previous process fails (that is, if the SLAT does not contain an entry for that IPA for the requesting guest), then a second-level page fault occurs.

- HM-6: The hypervisor traps second-level page faults. Upon such a page fault, the hypervisor will refuse the IPA request and will report this failure back to the guest (which may result in a "blue screen" on the guest). This corresponds to what has been described in requirement VM-7 for a single OS.

- HM-7: The SLAT of each guest is structured as a tree of pages with a root and two levels.

- HM-8: An IPA is a 32 bits word made of three parts: the level 1 part is made of the 10 upper bits of the IPA, the level 2 part is made of 10 intermediate bits, and the level 3 part is made of the 12 lower bits.

- HM-9: The level 1 bits of an IPA address the root page of the SLAT. This root page is itself pointed to by the base address of the SLAT (see HM-3). The root page is made of 1024 words. The contents of the word of this root page point to 1024 words level 2 pages.

- HM-10: A level 2 page is a word page of size 1024. The level 2 bits of an IPA address a word in a level 2 page. The contents of the word of this level 2 page point to 1024 words level 3 pages.

- HM-11: A level 3 page is a byte page of size 4096. The level 3 bits of an IPA address a byte in a level 3 page. A level 3 page of a SLAT is part of the memory of the guest associated with that SLAT.

- HM-12: The hypervisor is provided with a SLAT-TLB mapping some of the guest IPAs to the corresponding PAs. The SLAT-TLB is a short-circuit avoiding to always walk through the three level pages of the SLAT.

- HM-13: Once the hypervisor has walked through a SLAT in order to map an IPA to a PA, this pair is entered into the SLAT-TLB to be reused directly if another usage of that pair is needed. In doing so, a pair of the SLAT-TLB is evicted in order to make room for the new one.

1.6 Hypervisor scheduling

- HS-0: The hypervisor runs on a system providing several cores (CPUs).

- HS-1: Each core is provided with a physical MMU-TLB and a physical SLAT-TLB.

- HS-2: Each core has a physical base register containing the base address of a SLAT (see HM-3).

- HS-3: The number of guest controlled by the scheduler might be greater than the number of cores.

- HS-4: A guest that is not assigned to a core is said to be a sleeping guest.

- HS-5: The hypervisor regularly schedules a sleeping guest to a core. The guest currently running on the core is made sleeping. When scheduling, the MMU-TLB and the SLAT-TLB of that core are flushed.

- HS-6: When a guest is scheduled to a core, the base register of that core is updated with the base address of the SLAT of the scheduled guest (see requirement HM-3).

1.7 Hypervisor interrupt handling

- HI-0: The interrupt system of the hypervisor is virtualizing the interrupts of the individual guests when a physical interrupt occurs.

- HI-1: The guests will still believe that they are working on their individual machines with their own interrupts.

- HI-2: At boot time, an injective connection is established between each physical interrupt and a pair made of a guest and an interrupt of that guest.

- HI-3: Each physical interrupt is thus connected to a single virtual interrupt belonging to a guest.

- HI-4: When a physical interrupt occurs, it is made pending by the hardware, then made active when acknowledged by the hypervisor. After that, the hypervisor makes the corresponding virtual interrupt pending for the corresponding guest as if it were issued directly by the device connected to that guest.

- HI-5: When the scheduling of a guest occurs, its virtual core is attached to a physical core of the hypervisor.

- HI-6: An active physical interrupt is made effectively available to its guest as a virtual interrupt if that guest has been scheduled to a physical core. The pending virtual interrupt is then possibly taken into account by the guest as explained above in section 1.2. The state of the physical interrupt remains active.

- HI-7: An active physical interrupt remains active while the concerned guest is not scheduled to a physical core.

- HI-8: Once a virtual interrupt has been treated by its guest, an EOI signal is sent by the guest and the virtual interrupt is made inactive together with the corresponding physical interrupt.

- HI-9: Masking and unmasking of a virtual interrupt by a guest is made available to the hypervisor so that the corresponding physical interrupt can be masked (and made inactive if pending) or unmasked if already masked.