Internal ABI between the kernel and HYP

This file documents the interaction between the Linux kernel and the hypervisor layer when running Linux as a hypervisor (for example KVM). It doesn't cover the interaction of the kernel with the hypervisor when running as a guest (under Xen, KVM or any other hypervisor), or any hypervisor-specific interaction when the kernel is used as a host.

Note: KVM/arm has been removed from the kernel. The API described here is still valid though, as it allows the kernel to kexec when booted at HYP. It can also be used by a hypervisor other than KVM if necessary.

On arm and arm64 (without VHE), the kernel doesn't run in hypervisor mode, but still needs to interact with it, allowing a built-in hypervisor to be either installed or torn down.

In order to achieve this, the kernel must be booted at HYP (arm) or EL2 (arm64), allowing it to install a set of stubs before dropping to SVC/EL1. These stubs are accessible by using a 'hvc #0' instruction, and only act on individual CPUs.

Unless specified otherwise, any built-in hypervisor must implement these functions (see arch/arm{,64}/include/asm/virt.h):

-
  ```
  r0/x0 = HVC_SET_VECTORS
  r1/x1 = vectors
  ```

  Set HVBAR/VBAR_EL2 to 'vectors' to enable a hypervisor. 'vectors' must be a physical address, and respect the alignment requirements of the architecture. Only implemented by the initial stubs, not by Linux hypervisors.

-
  ```
  r0/x0 = HVC_RESET_VECTORS
  ```

  Turn HYP/EL2 MMU off, and reset HVBAR/VBAR_EL2 to the initials stubs' exception vector value. This effectively disables an existing hypervisor.

-
  ```
  r0/x0 = HVC_SOFT_RESTART
  r1/x1 = restart address
  x2 = x0's value when entering the next payload (arm64)
  x3 = x1's value when entering the next payload (arm64)
  x4 = x2's value when entering the next payload (arm64)
  ```

  Mask all exceptions, disable the MMU, clear I+D bits, move the arguments into place (arm64 only), and jump to the restart address while at HYP/EL2. This hypercall is not expected to return to its caller.

-
  ```
  x0 = HVC_VHE_RESTART (arm64 only)
  ```

  Attempt to upgrade the kernel's exception level from EL1 to EL2 by enabling the VHE mode. This is conditioned by the CPU supporting VHE, the EL2 MMU being off, and VHE not being disabled by any other means (command line option, for example).

Any other value of r0/x0 triggers a hypervisor-specific handling, which is not documented here.

The return value of a stub hypercall is held by r0/x0, and is 0 on success, and HVC_STUB_ERR on error. A stub hypercall is allowed to clobber any of the caller-saved registers (x0-x18 on arm64, r0-r3 and ip on arm). It is thus recommended to use a function call to perform the hypercall.