Kernel Memory Layout on ARM Linux

Russell King <rmk@arm.linux.org.uk>

November 17, 2005 (2.6.15)

This document describes the virtual memory layout which the Linux kernel uses for ARM processors. It indicates which regions are free for platforms to use, and which are used by generic code.

The ARM CPU is capable of addressing a maximum of 4GB virtual memory space, and this must be shared between user space processes, the kernel, and hardware devices.

As the ARM architecture matures, it becomes necessary to reserve certain regions of VM space for use for new facilities; therefore this document may reserve more VM space over time.

Start	End	Use
ffff8000	ffffffff	copy_user_page / clear_user_page use. For SA11xx and Xscale, this is used to setup a minicache mapping.
ffff4000	ffffffff	cache aliasing on ARMv6 and later CPUs.
ffff1000	ffff7fff	Reserved. Platforms must not use this address range.
ffff0000	ffff0fff	CPU vector page. The CPU vectors are mapped here if the CPU supports vector relocation (control register V bit.)
fffe0000	fffeffff	XScale cache flush area. This is used in proc-xscale.S to flush the whole data cache. Free for other usage on non-XScale.
fff00000	fffdffff	Fixmap mapping region. Addresses provided by fix_to_virt() will be located here.
ffc00000	ffefffff	DMA memory mapping region. Memory returned by the dma_alloc_xxx functions will be dynamically mapped here.
ff000000	ffbfffff	Reserved for future expansion of DMA mapping region.
VMALLOC_END	feffffff	Free for platform use, recommended. VMALLOC_END must be aligned to a 2MB boundary.
VMALLOC_START	VMALLOC_END-1	vmalloc() / ioremap() space. Memory returned by vmalloc/ioremap will be dynamically placed in this region. VMALLOC_START may be based upon the value of the high_memory variable.

PAGE_OFFSET	high_memory-1	Kernel direct-mapped RAM region. This maps the platforms RAM, and typically maps all platform RAM in a 1:1 relationship.
PKMAP_BASE	PAGE_OFFSET-1	Permanent kernel mappings One way of mapping HIGHMEM pages into kernel space.
MODULES_VADDR	MODULES_END-1	Kernel module space Kernel modules inserted via insmod are placed here using dynamic mappings.
00001000	TASK_SIZE-1	User space mappings Per-thread mappings are placed here via the mmap() system call.
00000000	00000fff	CPU vector page / null pointer trap CPUs which do not support vector remapping place their vector page here. NULL pointer dereferences by both the kernel and user space are also caught via this mapping.

Please note that mappings which collide with the above areas may result in a non-bootable kernel, or may cause the kernel to (eventually) panic at run time.

Since future CPUs may impact the kernel mapping layout, user programs must not access any memory which is not mapped inside their 0x0001000 to TASK_SIZE address range. If they wish to access these areas, they must set up their own mappings using open() and mmap().