Porting

Taken from list archive at http://lists.arm.linux.org.uk/pipermail/linux-arm-kernel/2001-July/004064.html

Initial definitions

The following symbol definitions rely on you knowing the translation that __virt_to_phys() does for your machine. This macro converts the passed virtual address to a physical address. Normally, it is simply:

phys = virt - PAGE_OFFSET + PHYS_OFFSET

Decompressor Symbols

ZTEXTADDR

Start address of decompressor. There's no point in talking about virtual or physical addresses here, since the MMU will be off at the time when you call the decompressor code. You normally call the kernel at this address to start it booting. This doesn't have to be located in RAM, it can be in flash or other read-only or read-write addressable medium.

ZBSSADDR

Start address of zero-initialised work area for the decompressor. This must be pointing at RAM. The decompressor will zero initialise this for you. Again, the MMU will be off.

ZRELADDR

This is the address where the decompressed kernel will be written, and eventually executed. The following constraint must be valid:

__virt_to_phys(TEXTADDR) == ZRELADDR

The initial part of the kernel is carefully coded to be position independent.

INITRD_PHYS

Physical address to place the initial RAM disk. Only relevant if you are using the bootpImage stuff (which only works on the old struct param_struct).

INITRD_VIRT

Virtual address of the initial RAM disk. The following constraint must be valid:

__virt_to_phys(INITRD_VIRT) == INITRD_PHYS

PARAMS_PHYS

Physical address of the struct param_struct or tag list, giving the kernel various parameters about its execution environment.

Kernel Symbols

PHYS_OFFSET

Physical start address of the first bank of RAM.

PAGE_OFFSET

Virtual start address of the first bank of RAM. During the kernel boot phase, virtual address PAGE_OFFSET will be mapped to physical address PHYS_OFFSET, along with any other mappings you supply. This should be the same value as TASK_SIZE.

TASK_SIZE

The maximum size of a user process in bytes. Since user space always starts at zero, this is the maximum address that a user process can access+1. The user space stack grows down from this address.

Any virtual address below TASK_SIZE is deemed to be user process area, and therefore managed dynamically on a process by process basis by the kernel. I'll call this the user segment.

Anything above TASK_SIZE is common to all processes. I'll call this the kernel segment.

(In other words, you can't put IO mappings below TASK_SIZE, and hence PAGE_OFFSET).

TEXTADDR

Virtual start address of kernel, normally PAGE_OFFSET + 0x8000. This is where the kernel image ends up. With the latest kernels, it must be located at 32768 bytes into a 128MB region. Previous kernels placed a restriction of 256MB here.

DATAADDR

Virtual address for the kernel data segment. Must not be defined when using the decompressor.

VMALLOC_START / VMALLOC_END

Virtual addresses bounding the vmalloc() area. There must not be any static mappings in this area; vmalloc will overwrite them. The addresses must also be in the kernel segment (see above). Normally, the vmalloc() area starts VMALLOC_OFFSET bytes above the last virtual RAM address (found using variable high_memory).

VMALLOC_OFFSET

Offset normally incorporated into VMALLOC_START to provide a hole between virtual RAM and the vmalloc area. We do this to allow out of bounds memory accesses (eg, something writing off the end of the mapped memory map) to be caught. Normally set to 8MB.

Architecture Specific Macros

BOOT_MEM(pram,pio,vio)

pram specifies the physical start address of RAM. Must always be present, and should be the same as PHYS_OFFSET.

pio is the physical address of an 8MB region containing IO for use with the debugging macros in arch/arm/kernel/debug-armv.S.

vio is the virtual address of the 8MB debugging region.

It is expected that the debugging region will be re-initialised by the architecture specific code later in the code (via the MAPIO function).

BOOT_PARAMS

Same as, and see PARAMS_PHYS.

FIXUP(func)

Machine specific fixups, run before memory subsystems have been initialised.

MAPIO(func)

Machine specific function to map IO areas (including the debug region above).

INITIRQ(func)

Machine specific function to initialise interrupts.