```
void * ioremap(unsigned long phys_addr, unsigned long size, unsigned long flags);
```

ioremap()的实现完全是平台相关的。

in arch/arm/include/asm/io.h

```
#define ioremap(cookie,size)          __arm_ioremap((cookie), (size), MT_DEVICE)
```

arch/arm/mm/ioremap.c

```
void __iomem * (*arch_ioremap_caller)(phys_addr_t, size_t,
                                      unsigned int, void *) =
        __arm_ioremap_caller;

void __iomem *
__arm_ioremap(phys_addr_t phys_addr, size_t size, unsigned int mtype)
{
        return arch_ioremap_caller(phys_addr, size, mtype,
                __builtin_return_address(0));
}
EXPORT_SYMBOL(__arm_ioremap);

void __iomem *__arm_ioremap_caller(phys_addr_t phys_addr, size_t size,
        unsigned int mtype, void *caller)
```

```
{
	phys_addr_t last_addr;
	unsigned long offset = phys_addr & ~PAGE_MASK;
	unsigned long pfn = __phys_to_pfn(phys_addr);

	/*
	 * Don't allow wraparound or zero size
	 */
	last_addr = phys_addr + size - 1;
	if (!size || last_addr < phys_addr)
		return NULL;

	return __arm_ioremap_pfn_caller(pfn, offset, size, mtype,
			caller);
}

void __iomem * __arm_ioremap_pfn_caller(unsigned long pfn,
	unsigned long offset, size_t size, unsigned int mtype, void *caller)
{
	const struct mem_type *type;
	int err;
	unsigned long addr;
	struct vm_struct *area;
	phys_addr_t paddr = __pfn_to_phys(pfn);
```

```
#ifndef CONFIG_ARM_LPAE

        /*

         * High mappings must be supersection aligned

         */

        if (pfn >= 0x100000 && (paddr & ~SUPERSECTION_MASK))    检查page frame是否在4G以外
了

                return NULL;

#endif



        type = get_mem_type(mtype);

        if (!type)

                return NULL;



        /*

         * Page align the mapping size, taking account of any offset.

         */

        size = PAGE_ALIGN(offset + size);



        /*

         * Try to reuse one of the static mapping whenever possible.

         */

        if (size && !(sizeof(phys_addr_t) == 4 && pfn >= 0x100000)) {

                struct static_vm *svm;



                svm = find_static_vm_paddr(paddr, size, mtype);         ( 1 )
                if (svm) {
                        addr = (unsigned long)svm->vm.addr;
                        addr += paddr - svm->vm.phys_addr;
                        return (void __iomem *) (offset + addr);
                }
        }

        /*
         * Don't allow RAM to be mapped - this causes problems with ARMv6+
         */
        if (WARN_ON(pfn_valid(pfn)))
                return NULL;

        area = get_vm_area_caller(size, VM_IOREMAP, caller);     ( 2 )
        if (!area)
                return NULL;
        addr = (unsigned long)area->addr;
        area->phys_addr = paddr;

#if !defined(CONFIG_SMP) && !defined(CONFIG_ARM_LPAE)
        if (DOMAIN_IO == 0 &&
                (((cpu_architecture() >= CPU_ARCH_ARMv6) && (get_cr() & CR_XP)) ||
                  cpu_is_xsc3()) && pfn >= 0x100000 &&
                  !((paddr | size | addr) & ~SUPERSECTION_MASK)) {
                area->flags |= VM_ARM_SECTION_MAPPING;
```

```
                err = remap_area_supersections(addr, pfn, size, type);
        } else if (!((paddr | size | addr) & ~PMD_MASK)) {
                area->flags |= VM_ARM_SECTION_MAPPING;
                err = remap_area_sections(addr, pfn, size, type);
        } else
#endif
                err = ioremap_page_range(addr, addr + size, paddr,      （3）
                                         __pgprot(type->prot_pte));

        if (err) {
                vunmap((void *)addr);
                return NULL;
        }

        flush_cache_vmap(addr, addr + size);
        return (void __iomem *) (offset + addr);
}
```

（1）find_static_vm_paddr()

在ARM setup_arch() / setup.c中对某些devic的mapping address会挂在static_vmlist list上。

该list上是static mapping的io address（如下）。

```
static struct map_desc pegmatite_io_desc[] __initdata = {
        {
```

```
		.virtual	= (unsigned long) PEGMATITE_REGS_VIRT_BASE,
		.pfn		= __phys_to_pfn(PEGMATITE_REGS_PHYS_BASE),
		.length		= PEGMATITE_REGS_SIZE,
		.type		= MT_DEVICE,
	},
	{
		.virtual	= (unsigned long) PEGMATITE_UPC_VIRT_BASE,
		.pfn		= __phys_to_pfn(PEGMATITE_UPC_PHYS_BASE),
		.length	= 0x000C0000,
		.type		= MT_DEVICE
	},
};

void __init pegmatite_map_io(void)
{
	iotable_init(pegmatite_io_desc, ARRAY_SIZE(pegmatite_io_desc));
}
```

find_static_vm_paddr()搜索在static_vmlist上的已有的mapping address，看是否当前要ioremap的address已经mapping了。

如果是，则直接返回即可。

（2）get_vm_area_caller()

```
struct vm_struct *get_vm_area_caller(unsigned long size, unsigned long flags,
				const void *caller)
```

```
{
        return __get_vm_area_node(size, 1, flags, VMALLOC_START, VMALLOC_END,
                                NUMA_NO_NODE, GFP_KERNEL, caller);
}
```

在[VMALLOC_START, VMALLOC_END),即0xf0000000 - 0xff000000之间找寻一段size的地址空间，get_vm_area_caller（ ）返回值就是代表这段虚拟空间的vm_struct。

从/proc/vmallocinfo文件可验证ioremap()建立的 virtual address v.s. physical address之间的mapping。

```
root@granite2:~# cat /proc/vmallocinfo
0xbf000000-0xbf002000    8192 module_alloc_update_bounds+0xc/0x5c pages=1 vmalloc
0xbf004000-0xbf007000   12288 module_alloc_update_bounds+0xc/0x5c pages=2 vmalloc
0xbf009000-0xbf00d000   16384 module_alloc_update_bounds+0xc/0x5c pages=3 vmalloc
0xbf02c000-0xbf02f000   12288 module_alloc_update_bounds+0xc/0x5c pages=2 vmalloc
0xbf031000-0xbf038000   28672 module_alloc_update_bounds+0xc/0x5c pages=6 vmalloc
0xbf03b000-0xbf03e000   12288 module_alloc_update_bounds+0xc/0x5c pages=2 vmalloc
0xbf047000-0xbf053000   49152 module_alloc_update_bounds+0xc/0x5c pages=11 vmallo
c

......

0xf0000000-0xf0002000    8192 of_iomap+0x30/0x38 phys=d1d01000 ioremap
0xf0002000-0xf0004000    8192 of_iomap+0x30/0x38 phys=d1d02000 ioremap
0xf0004000-0xf0007000   12288 of_iomap+0x30/0x38 phys=d0620000 ioremap
0xf0008000-0xf000a000    8192 of_iomap+0x30/0x38 phys=d0621000 ioremap
0xf000a000-0xf000c000    8192 of_iomap+0x30/0x38 phys=d0627000 ioremap
0xf000c000-0xf000e000    8192 of_iomap+0x30/0x38 phys=d0622000 ioremap
0xf000e000-0xf0010000    8192 of_iomap+0x30/0x38 phys=d0622000 ioremap
0xf0010000-0xf0012000    8192 of_iomap+0x30/0x38 phys=d0623000 ioremap
0xf0012000-0xf0014000    8192 of_iomap+0x30/0x38 phys=d0623000 ioremap
0xf0014000-0xf0016000    8192 of_iomap+0x30/0x38 phys=d0625000 ioremap

......
```

（3）ioremap_page_range()

```
int ioremap_page_range(unsigned long addr,
                unsigned long end, phys_addr_t phys_addr, pgprot_t prot)
```

```
{
	pgd_t *pgd;
	unsigned long start;
	unsigned long next;
	int err;

	BUG_ON(addr >= end);

	start = addr;
	phys_addr -= addr;
	pgd = pgd_offset_k(addr);
	do {
		next = pgd_addr_end(addr, end);
		err = ioremap_pud_range(pgd, addr, next, phys_addr+addr, prot);
		if (err)
			break;
	} while (pgd++, addr = next, addr != end);

	flush_cache_vmap(start, end);

	return err;
}
```

建立virtual address to physical page的页表。