include/linux/dma-mapping.h

arch/arm/include/asm/dma-mapping.h

include/asm-generic/dma-mapping-common.h

这3者的include关系?

使用dma的driver需要include如下头文件

```
1.  #include <linux/dma-mapping.h>
2.  #include <linux/dmaengine.h>
```

由于cdma driver的特殊性(其实我觉得是code实现上的问题)，还需要

```
1.  #include <linux/platform_data/mv61_cdma.h>
```

#include <linux/dmaengine.h>是为了access dmaengine APIs.

#include <linux/dma-mapping.h>则是为了access dma memory allocation APIs.

定义在linux/dma-mapping.h中的struct dma_map_ops就象是c++中的base class，而arch/arm/include/asm/dma-mapping.h则作为其的子类实现了struct dma_map_ops中的纯虚函数。每个arch都是linux/dma-mapping.h的一个子类，都必须实现自己的struct dma_map_ops中的callback。

而include/asm-generic/dma-mapping-common.h则作为所有arch都共通的部分被抽取出来。

```
struct dma_map_ops {
        void* (*alloc)(struct device *dev, size_t size,
                               dma_addr_t *dma_handle, gfp_t gfp,
                               struct dma_attrs *attrs);
        void (*free)(struct device *dev, size_t size,
                              void *vaddr, dma_addr_t dma_handle,
                              struct dma_attrs *attrs);
        int (*mmap)(struct device *, struct vm_area_struct *,
                            void *, dma_addr_t, size_t, struct dma_attrs *attrs);

        int (*get_sgtable)(struct device *dev, struct sg_table *sgt, void *,
                            dma_addr_t, size_t, struct dma_attrs *attrs);

        dma_addr_t (*map_page)(struct device *dev, struct page *page,
                               unsigned long offset, size_t size,
                               enum dma_data_direction dir,
                               struct dma_attrs *attrs);
        void (*unmap_page)(struct device *dev, dma_addr_t dma_handle,
                            size_t size, enum dma_data_direction dir,
                            struct dma_attrs *attrs);
        int (*map_sg)(struct device *dev, struct scatterlist *sg,
                      int nents, enum dma_data_direction dir,
                      struct dma_attrs *attrs);
        void (*unmap_sg)(struct device *dev,
                         struct scatterlist *sg, int nents,
                         enum dma_data_direction dir,
                         struct dma_attrs *attrs);
        void (*sync_single_for_cpu)(struct device *dev,
                                    dma_addr_t dma_handle, size_t size,
                                    enum dma_data_direction dir);
        void (*sync_single_for_device)(struct device *dev,
                                       dma_addr_t dma_handle, size_t size,
                                       enum dma_data_direction dir);
        void (*sync_sg_for_cpu)(struct device *dev,
                                struct scatterlist *sg, int nents,
                                enum dma_data_direction dir);
        void (*sync_sg_for_device)(struct device *dev,
                                   struct scatterlist *sg, int nents,
                                   enum dma_data_direction dir);
        int (*mapping_error)(struct device *dev, dma_addr_t dma_addr);
        int (*dma_supported)(struct device *dev, u64 mask);
        int (*set_dma_mask)(struct device *dev, u64 mask);
#ifdef ARCH_HAS_DMA_GET_REQUIRED_MASK
        u64 (*get_required_mask)(struct device *dev);
#endif
        int is_phys;
};
```

arch/arm/include/asm/dma-mapping.h

arch/arm/include/asm/dma-mapping.c

```
struct dma_map_ops arm_dma_ops = {
        .alloc                  = arm_dma_alloc,
        .free                   = arm_dma_free,
        .mmap                   = arm_dma_mmap,
        .get_sgtable            = arm_dma_get_sgtable,
        .map_page               = arm_dma_map_page,
        .unmap_page             = arm_dma_unmap_page,
        .map_sg                 = arm_dma_map_sg,
        .unmap_sg               = arm_dma_unmap_sg,
        .sync_single_for_cpu    = arm_dma_sync_single_for_cpu,
        .sync_single_for_device = arm_dma_sync_single_for_device,
        .sync_sg_for_cpu        = arm_dma_sync_sg_for_cpu,
        .sync_sg_for_device     = arm_dma_sync_sg_for_device,
        .set_dma_mask           = arm_dma_set_mask,
};
```

```
struct dma_map_ops arm_coherent_dma_ops = {
        .alloc                  = arm_coherent_dma_alloc,
        .free                   = arm_coherent_dma_free,
        .mmap                   = arm_dma_mmap,
        .get_sgtable            = arm_dma_get_sgtable,
        .map_page               = arm_coherent_dma_map_page,
        .map_sg                 = arm_dma_map_sg,
        .set_dma_mask           = arm_dma_set_mask,
};
```

比如

```
#define dma_alloc_coherent(d, s, h, f) dma_alloc_attrs(d, s, h, f, NULL)
```

```
static inline void *dma_alloc_attrs(struct device *dev, size_t size,
                                    dma_addr_t *dma_handle, gfp_t flag,
                                    struct dma_attrs *attrs)
{
        struct dma_map_ops *ops = get_dma_ops(dev);
        void *cpu_addr;
        BUG_ON(!ops);

        cpu_addr = ops->alloc(dev, size, dma_handle, flag, attrs);
        debug_dma_alloc_coherent(dev, size, *dma_handle, cpu_addr);
        return cpu_addr;
}
```

dma_alloc_attrs()是arch-neutral，但具体实现确实依赖于arch的。