1. **概念**

内存空间：内存的所占用的地址空间

Io空间：独立于内存空间的的一段地址空间，供io寄存器使用

Io寄存器：每个设备都会提供几个寄存器，这些寄存器的地址是连续的，通过对寄存器的读写控制设备，有如下几类寄存器

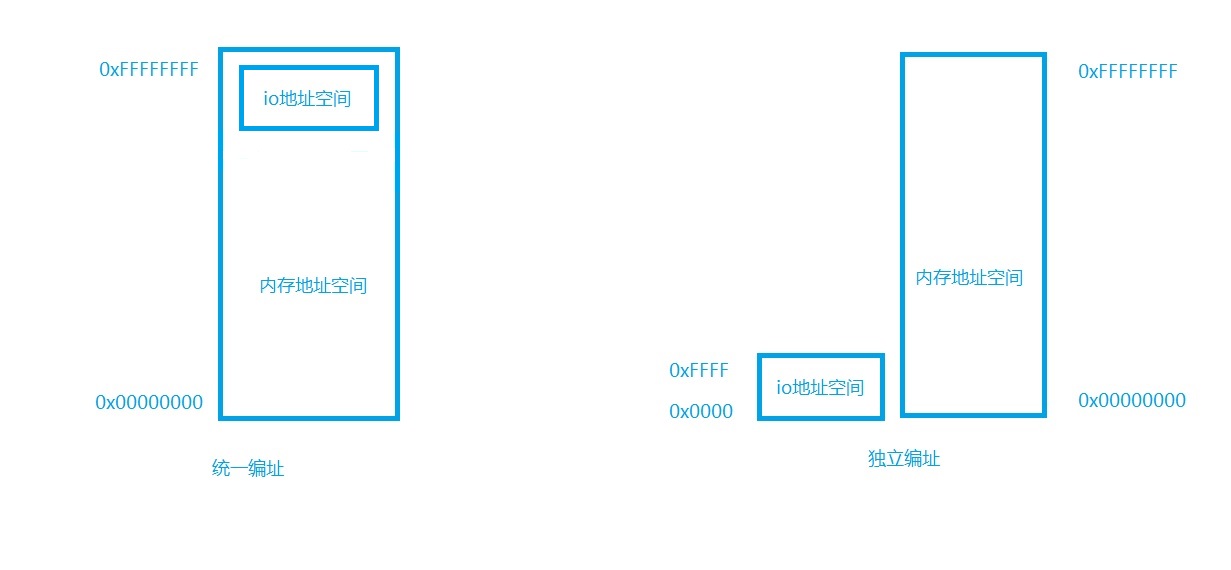
数据寄存器：通过读写该寄存器完成对设备的读写

状态寄存器：设备的状态

控制寄存器：控制设备的行为

Io端口：当io寄存器地址使用独立编址方式时，称为io端口

Io内存：当io寄存器或设备内存使用统一编址方式时，称为io内存



1. **Io寄存器与常规内存**

边际效应：读取某个地址的值后，他的值会发生变化

对内存的读写，不存在边界效应

对io寄存器的读写存在边际效应

编译器的代码优化会调整汇编指令的读写顺序，这对读写内存每什么影响，但对读写io寄存器有很大影响，因为io寄存器存在边际效应

解决如上问题的方法是在读写寄存器的代码中加入内存屏障

Linux 提供 4 个宏

#include <linux/kernel.h>

void barrier(void)      // 禁止这段代码的前后代码之间的优化

#include <asm/system.h>

void rmb(void);         // 保证这段代码之前的读操作执行完成后在执行其后代码

void wmb(void);         // 保证这段代码之前的写操作执行完成后在执行其后代码

void mb(void);          // 保证这段代码之前写的读操作执行完成后在执行其后代码

1. **读写io端口**

**Io端口分配**

在对端口操作之前，应该告诉内核我们需要占用这些端口

#include <linux/ioport.h>

struct resource \*request\_region(unsigned long first, unsigned long n,

const char \*name);

First：要占用的端口的开始地址（物理地址）

N：占用多少个

Name：设备的名称

返回值：如果分配成功返回值是非 NULL

**操作io端口**

8位端口读写

unsigned inb(unsigned port);

void outb(unsigned char byte, unsigned port);

16位端口读写

unsigned inw(unsigned port);

void outw(unsigned short word, unsigned port);

32位端口读写

unsigned inl(unsigned port);

void outl(unsigned longword, unsigned port);

Port为设备物理地址对应的虚拟地址，将物理地址映射到虚拟地址使用ioremap，下面将会讲到

**io端口释放**

当我们不在需要端口时，应该释放

void release\_region(unsigned long start, unsigned long n);

1. **读写io内存**

**请求占用内存区域**

struct resource \*request\_mem\_region(unsigned long start, unsigned long

len, char \*name);

Start：设备的物理地址

Len：请求内存区域的长度

Name：设备名

返回值：成功返回非null的指针

**映射内存区域**

我们无法直接使用物理地址，应该映射为虚拟地址后在使用

#include <asm/io.h>

void \*ioremap(unsigned long phys\_addr, unsigned long size);     // 映射物理地址，返回一个虚拟地址

**操作io内存**

映射之后，我们可以直接操作返回的虚拟地址的指针，但为了可移植性，我们应该使用下面的函数

从 I/O 内存读, 使用下列之一:

unsigned int ioread8(void \*addr);

unsigned int ioread16(void \*addr);

unsigned int ioread32(void \*addr);

从io内存写，使用下列函数

void iowrite8(u8 value, void \*addr);

void iowrite16(u16 value, void \*addr);

void iowrite32(u32 value, void \*addr);

**解除内存区域映射**

#include <asm/io.h>

void iounmap(void \* addr);                                      // 解除的映射

**解除占用内存区域**

void release\_mem\_region(unsigned long start, unsigned long len);

Start：设备的物理地址

Len：请求内存区域的长度