嵌入式系统设计与应用

第二章 ARM处理器组成结构（2）

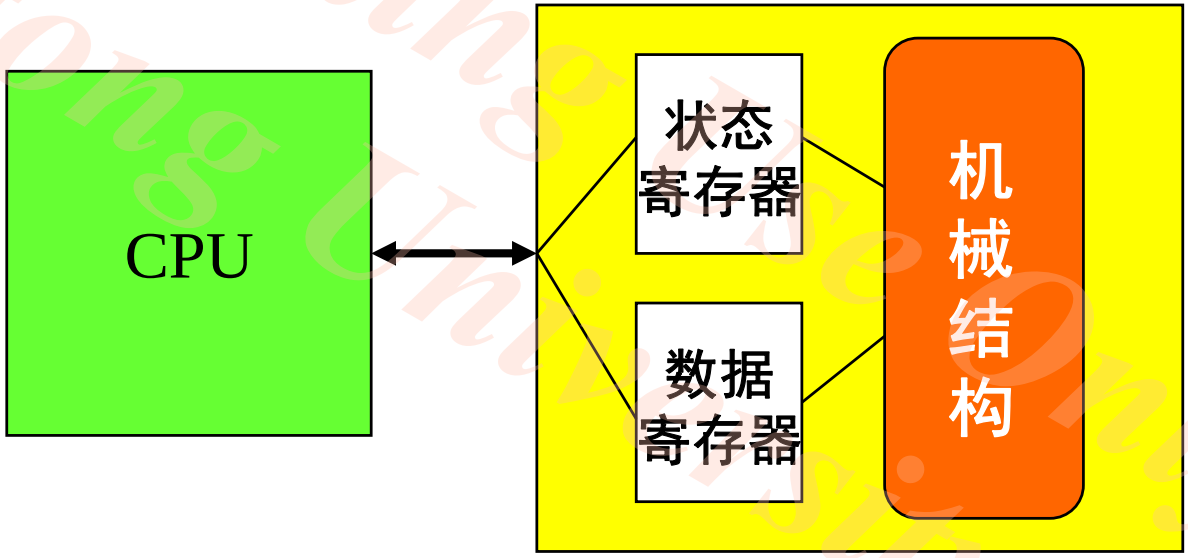
西安交通大学电信学院
孙宏滨

Hongbin Sun
Internal Teaching Use Only
Xi'an Jiaotong University

1 输入/输出编程

- **输入输出设备（I/O Devices）：**通常有一些模拟元件或非电子元件（比如磁盘驱动器）。
- 但与CPU的接口通常是数字接口---一组寄存器。
  - 数据寄存器：通过设备将这些值作为数据保存，如磁盘读写
  - 状态寄存器：提供设备运行的信息，如当前时间完成查询

寄存器可以是只读的，也可以是可读可写的。

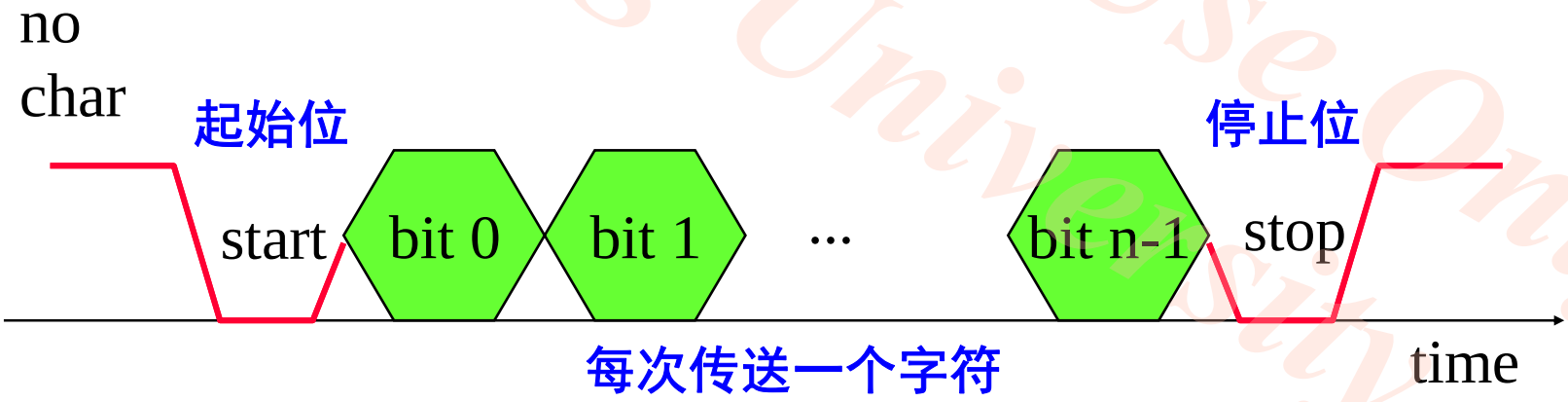


Honorin Sun Internal Teaching Use Only Xi'an Jiaotong University

应用示例：8251 UART

- **8251 UART(Universal Asynchronous Receiver and Transmitter)**通用异步收发器进行串行通信。
- **PC**的串口，早期独立芯片，现集成在较大的芯片中。
  - 波特率
  - 每个字符的位数（**5**到**8**）
  - 是否包含奇偶校验，是奇校验还是偶校验
  - 结束位的长度（**1**、**1.5**、还是**2**位）

  } 参数设置



Honqin Sun
International Teaching Use Only
Xi'an Jiaotong University

输入输出编程支持

- **I/O指令：** 如x86为I/O提供了特殊指令（in和out），这些指令为I/O设备提供了单独的地址空间。
- **内存映射I/O：** 最普遍的方法，即使提供I/O指令的CPU也不排除使用内存映射I/O。
  - 内存映射I/O是为每个I/O设备的寄存器提供一个地址。
  - 程序使用普通的CPU读写命令来与设备通信。

Honeywell Software Integration Laboratory Xi'an Jiaotong University For Use Only

ARM内存映射I/O

- 定义I/O设备地址:

```
DEV1 EQU 0x1000
```

- 读写设备寄存器代码:

```
LDR r1,#DEV1 ; set up device adrs
LDR r0,[r1]    ; read DEV1
LDR r0,#8      ; set up value to write
STR r0,[r1]    ; write value to device
```

Hongjun Sun
Internal Teaching Use Only
Xi'an Jiaotong University

高级语言直接写I/O设备

- 在C语言中定义和使用一个变量时，编译器隐藏了变量地址。可以使用指针来操纵I/O设备的地址。

```
int peek(char *location) { return *location; }
#define DEV1 0x1000
……
dev_status = peek(DEV1);  /*读设备寄存器*/

void poke(char *location, char newval) {
   (*location) = newval; }
poke(DEV1, 8);                /*写设备寄存器*/
```

Honeywell Sun Interwoven Teaching Use Only Xi'an Jiaotong University

忙等I/O

- 忙等I/O是程序中访问设备的最基本方法。
- 设备一般比CPU慢，需要多周期来完成一项操作。
- 通过读I/O设备状态寄存器来询问I/O设备是否空闲通常被称为轮询（polling）。

Hongbin Sun
Internal Teaching Use Only
Xi'an Jiaotong University

忙等I/O编程（1）

```
#define  OUT_CHAR    0x1000
#define  OUT_STATUS  0x1001
char  *mystring = "Hello world."
char  *current_char;
      current_char = mystring;
      while (*current_char != '\0') {
            poke(OUT_CHAR,*current_char);
            while (peek(OUT_STATUS) != 0);
            current_char++;
            }
```

Honqin Sun
Internal Teaching Use Only
Xi'an Jiaotong University

忙等I/O编程（2）

```
#define  IN_DATA     0x1000
#define  IN_STATUS   0x1001
#define  OUT_DATA    0x1100
#define  OUT_STATUS 0x1101
while (TRUE) {
   /* read */
   while (peek(IN_STATUS) == 0);
   achar = (char)peek(IN_DATA);
   /* write */
   poke(OUT_DATA,achar);
   poke(OUT_STATUS,1);
   while (peek(OUT_STATUS) != 0);
   }
```

Hongbin Sun
Internal Teaching Use Only
Xi'an Jiaotong University

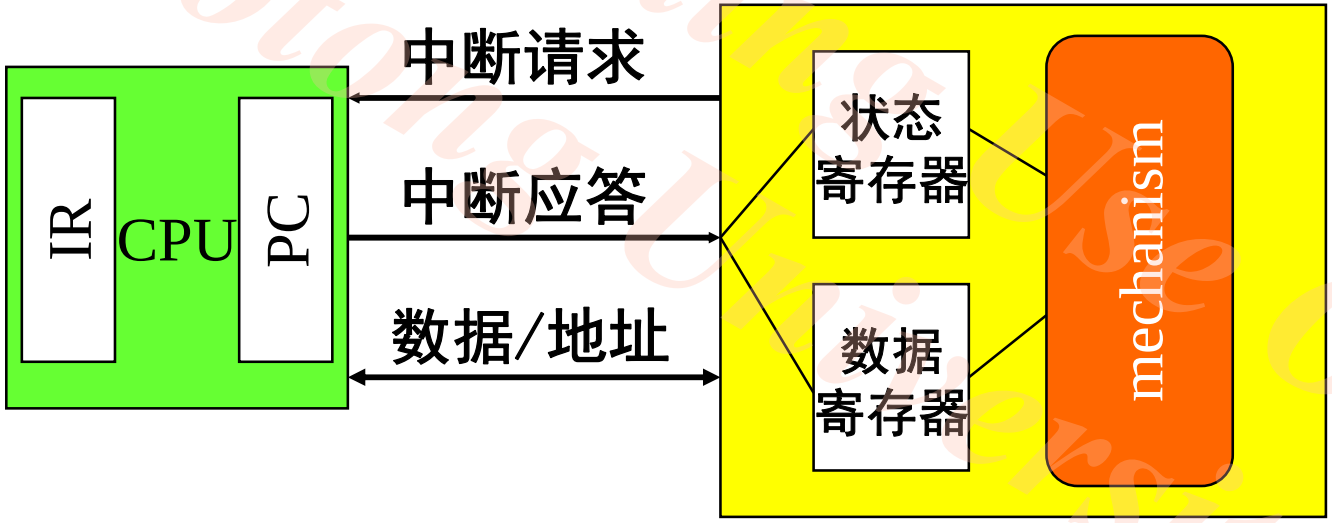
中断I/O

- 忙等I/O的效率非常低：
  - I/O事务未完成时，CPU除测试设备状态什么都不能做。
  - 无法将其他操作与I/O事务并行执行。
- 中断机制能允许设备发送信号到CPU，强制执行一段特定代码。
  - 中断发生时，PC指向I/O设备中一个管理设备的中断处理子程序（中断服务程序或设备驱动程序）。
  - 中断机制保留了系统被中断时的PC值，以便CPU能够返回到被中断的程序。
- 中断机制使得CPU能够在不同的上下文中间切换。

Hongbin Sun
Internal Teaching Use Only
Xi'an Jiaotong University

中断I/O接口

- 当需要CPU服务时，I/O设备就发送**中断请求**信号。
- 当CPU准备处理I/O设备的请求时，CPU便发送**中断应答**信号。



Honorin Study Internal Teaching Use Only Xi'an Jiaotong University

中断处理范例（1）

输入输出处理程序

```
void input_handler() {
   achar = peek(IN_DATA);
   gotchar = TRUE;
   poke(IN_STATUS,0);
}

void output_handler() {
}
```

主程序

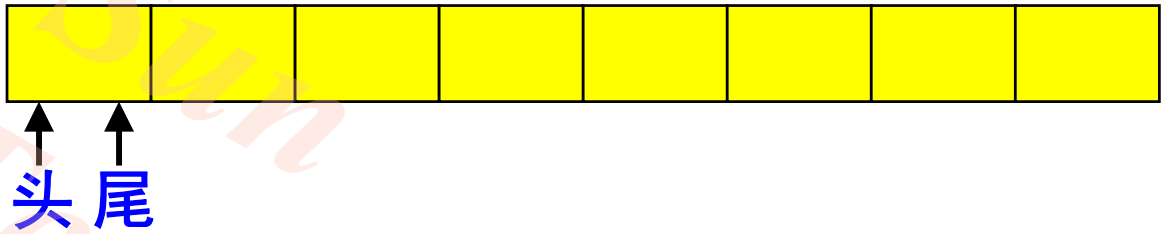
```
main() {
   while (TRUE) {
      if (gotchar) {
      poke(OUT_DATA,achar);
      poke(OUT_STATUS,1)
      While(
      peek(OUT_STATUS)!=0)
      gotchar = FALSE;
      }
   }
}
```

中断的使用使主程序稍微变简单，但是仍然没有让前台程序做有用的工作。

Honeywell Sun Internal Teaching Use Only Xi'an Jiaotong University

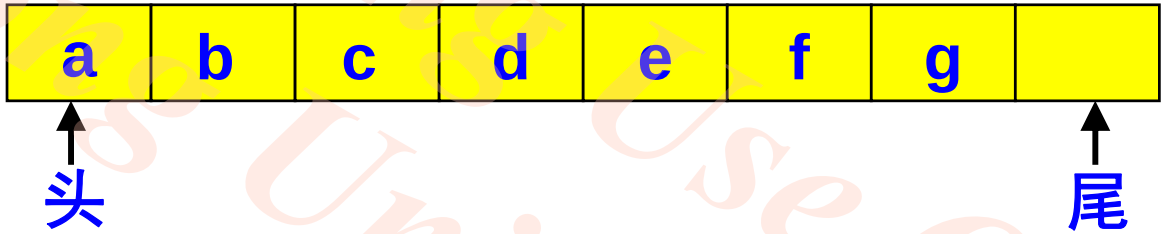
中断处理范例（2）- 1

队列为空

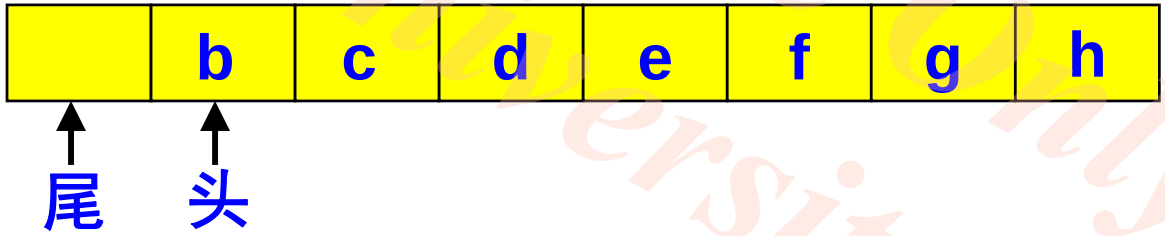


读入第一个字符

缓冲区满，最后一个字符位不使用



循环缓冲区



Honeywell Sun Internal Teaching Use Only Xi'an Haotong University

中断处理范例（2）- 2

```
#define BUF_SIZE 8
char io_buf[BUF_SIZE];
int   buf_head = 0, buf_tail = 0;
int   error = 0;
bool empty_buffer() {retun buf_head == buf_tail;}
bool full_buffer() {
        return (buf_tail+1)%BUF_SIZE == buf_head;}
int   nchars(){
        if(buf_tail>=buf_head) return buf_tail-buf_head;
        else return BUF_SIZE + buf_tail - buf_head;
}
```

Hongjun Sun
Interrupt Teaching Use Only
Xi'an Jiaotong University

中断处理范例（2）- 3

```
void add_char(char achar) {
   io_buf[buf_tail++] = achar;
   if(buf_tail == BUF_SIZE)
       buf_tail = 0;
}
char remove_char() {
   char achar;
   achar = io_buf[buf_head++];
   if(buf_head == BUF_SIZE)
       buf_head = 0;
}
```

Honeywell Solve Internal Teaching Use Only Xi'an Jiaotong University

中断处理范例（2）- 4

```
#define IN_DATA    0x1000
#define IN_STATUS 0x1001
void input_handler() {
   char achar;
   if (full_buffer()) error = 1;    /*出错*/
   else {                  achar = peek(IN_DATA);
                           add_char(achar);
   }
   poke(IN_STATUS,0);
   if (nchars == 1){ poke(OUT_DATA,remove_char());
                           poke(OUT_STATUS,1); }
}
```

Honeywin Sun
Internal Teaching Use Only
Xi'an Jiaotong University

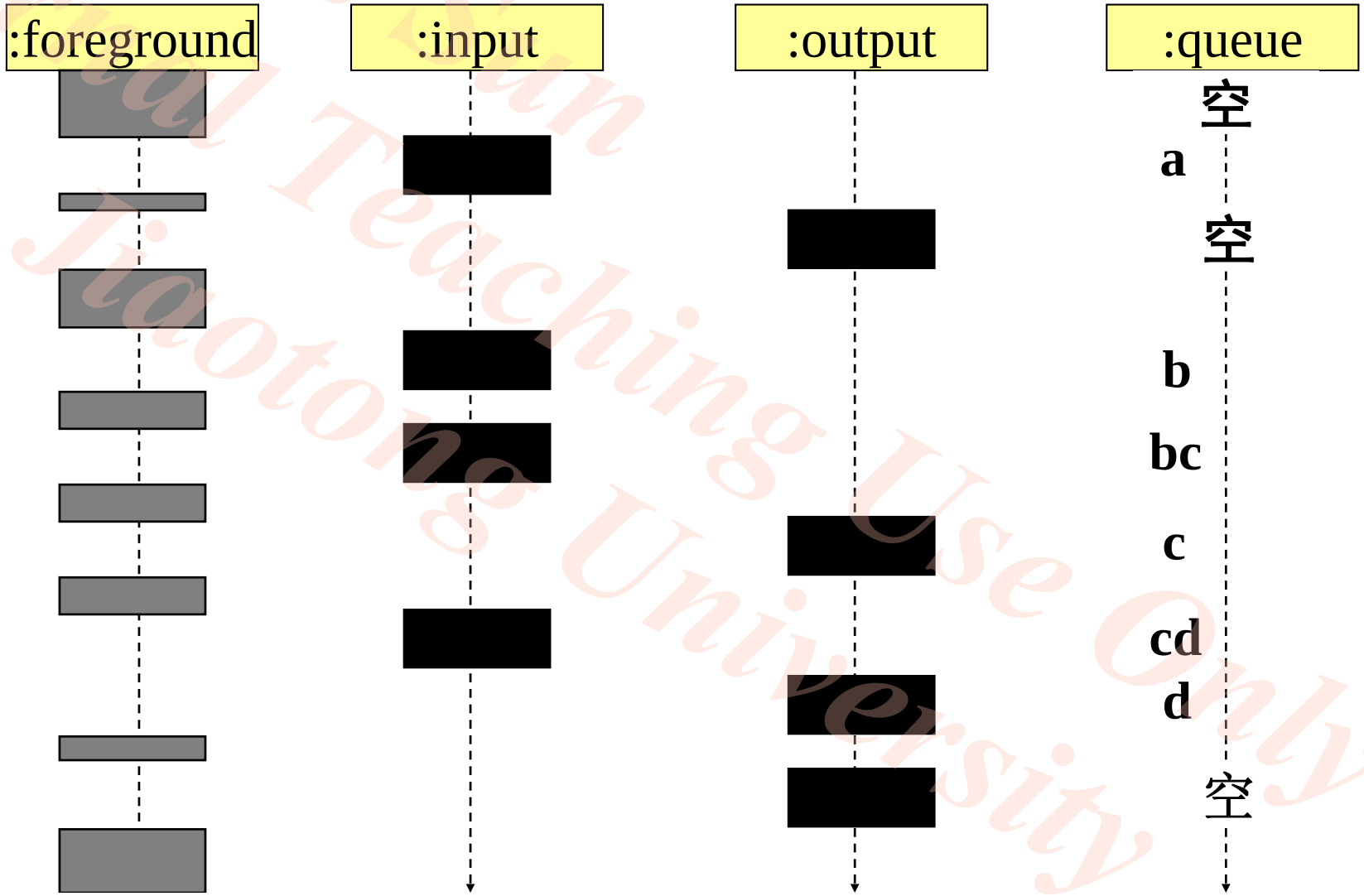
中断处理范例（2）- 5

```
#define OUT_DATA    0x1100
#define OUT_STATUS  0x1101
void output_handler() {
        if(!empty_buffer())
                poke(OUT_DATA, remove_char());
                poke(OUT_STATUS, 1);
        }
```

Honeywell Sun Internal Teaching Use Only
Xi'an Jiaotong University

中断处理范例（2）- 6

所有任务都由中断程序处理，前台程序可以完成其他任务或处于空闲。



Honeyun Sun
Internal Teaching Use Only
Xi'an Jiaotong University

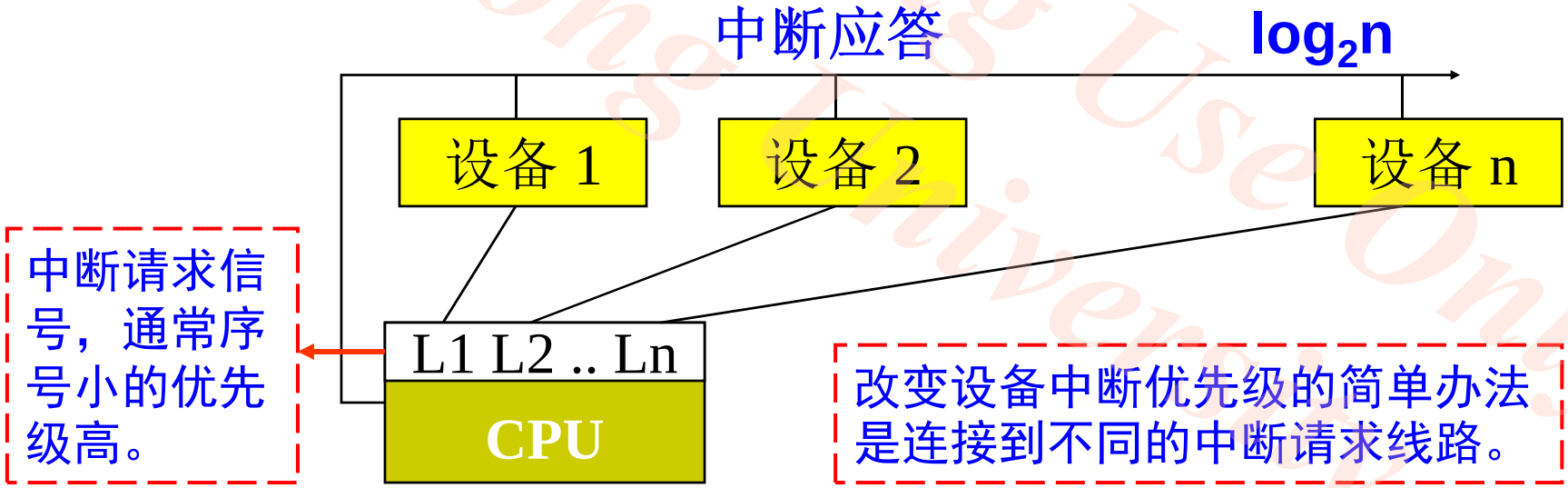
中断调试代码

- 在中断处理程序中，任何一个寄存器在被改写前必须保存原来的值，并且在中断处理程序结束前恢复其原来的值。
- 任何类型的错误，如忘记保存寄存器值或是没有正确恢复原值，都可能使前台程序的寄存器意想不到的被改变，产生错误。
- 由中断处理程序造成的前台程序出错很难调试：错误的形式取决于中断发生的时间，不具有重复性。

Honeywell Sure Internal Teaching Use Only Xi'an Jiaotong University

优先级和向量

- 有两种方法可以使中断通用化来处理多个设备，并且提供更灵活的相关软件和硬件的定义：
  - 中断优先级（**interrupt priority**）：能让**CPU**辨认出比其他中断更加重要的一些中断。
  - 中断向量（**interrupt vector**）：允许中断设备指定其中断处理程序。



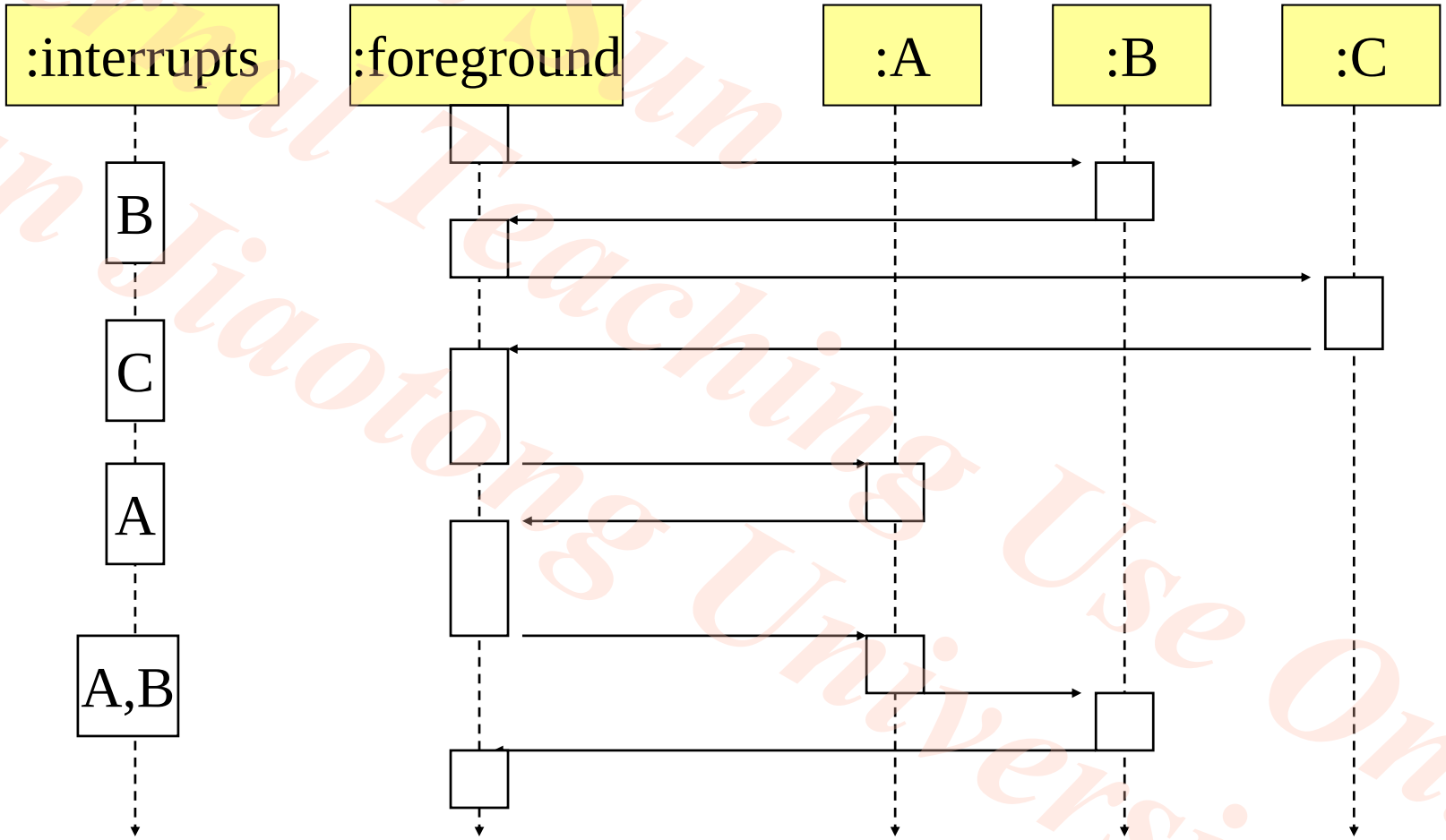
Honeywell Sensing & Interface Technology Xi'an Jiaotong University Use Only

中断优先级

- 屏蔽（**masking**）：优先级机制必须确保处理高优先级中断时，不发生低优先级中断。
  - 中断应答时，该中断的优先级存储在优先级寄存器中。
  - 接收到后续中断时，**CPU**会将该中断优先级与寄存器中的值比较，仅当优先级比最近挂起中断高时才应答。
  - 中断程序退出时，优先级寄存器必须被重置（特许中断返回指令）。
- 不可屏蔽中断（**nonmaskable interrupt, NMI**）：**NMI**不能断开，它通常是为由电源故障引发的中断准备的，是一个可以监测危险低电压的简单电路。

Hongjun Shu
Internal Teaching Use Only
Xi'an Jiaotong University

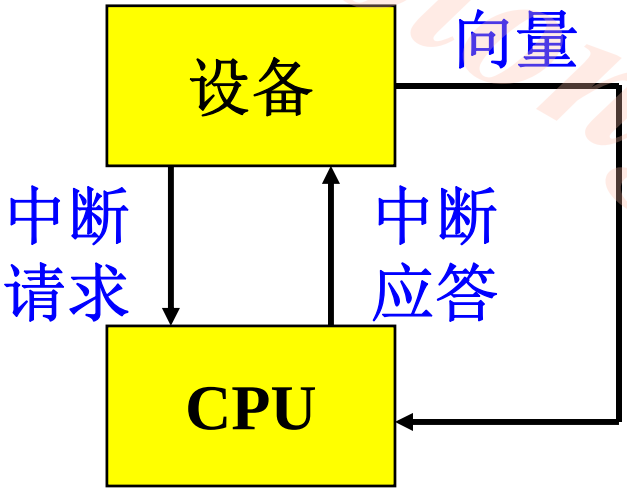
中断优先级示例



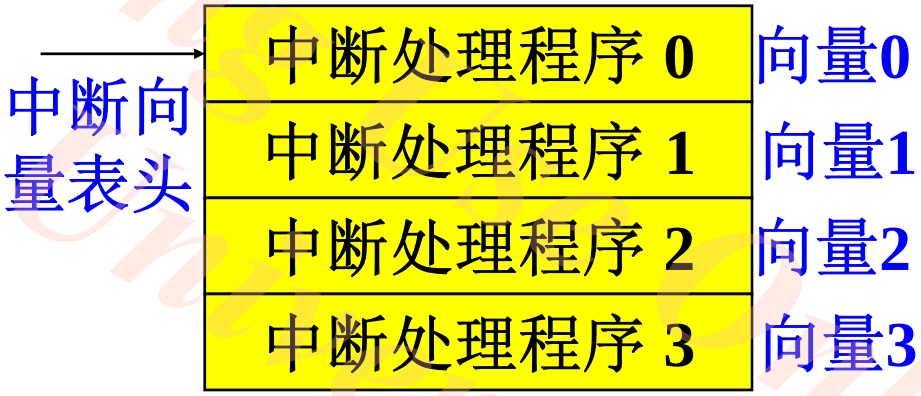
Hongqun Sun
Internal Teaching Use Only
Xi'an Jiaotong University

中断向量

- 在设备的请求或应答之后，它通过这些线路将中断向量发送到CPU。
- CPU将向量号作为存储在内存中的中断向量表的索引。
- 设备端存储自己的向量号，提供**灵活性**。



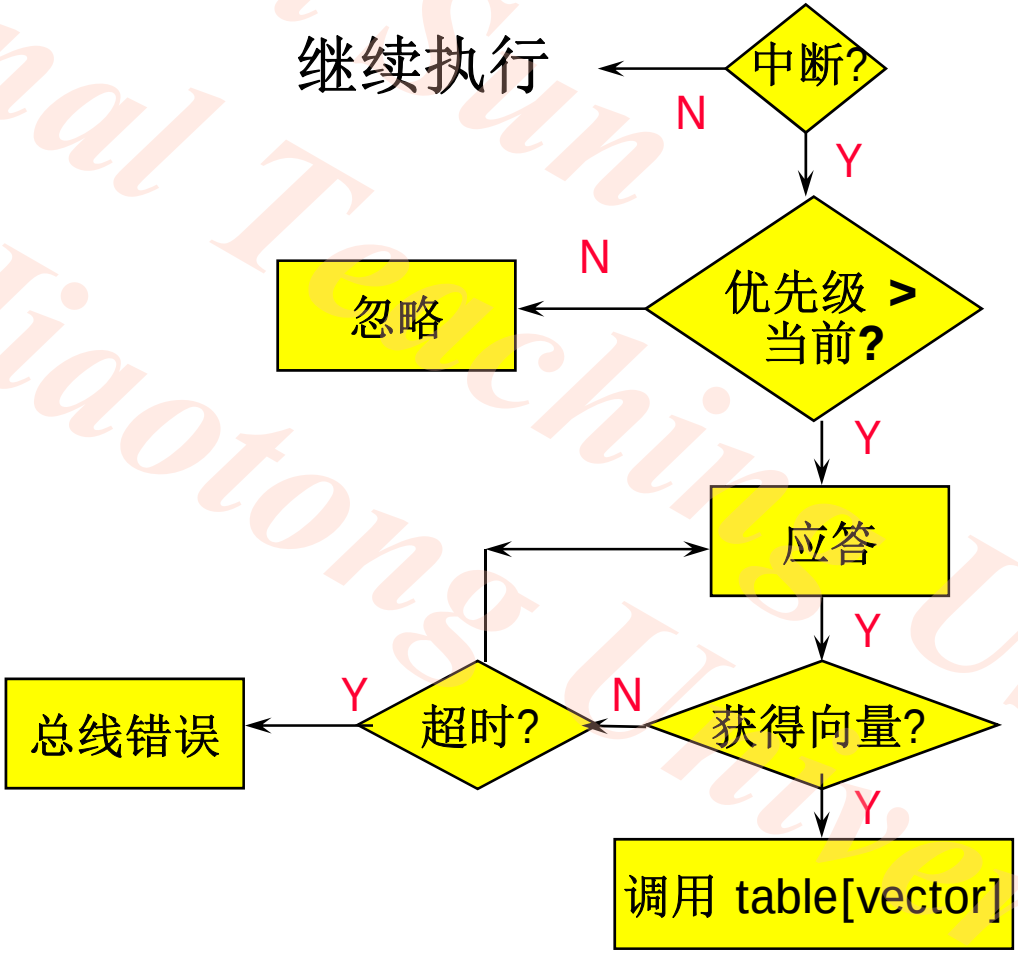
硬件结构



中断向量表

Hongqin Sun
Internet Technology
Xi'an Jiaotong University

通用的中断机制



Hongqun Sun
Internal Teaching Use Only
Xi'an Jiaotong University

中断过程序列

- 中断的主要过程步骤如下：
  - **CPU**：在每条指令的开始检查未决中断，并响应优先级最高的中断（高于优先级寄存器）。
  - 设备：接收中断应答信号并向**CPU**发送它的中断向量。
  - **CPU**：用向量作为索引在中断向量表中查找中断处理程序的地址。用一种类似子程序的机制保存**PC**和状态。
  - 软件：设备驱动程序将保存附加**CPU**状态，执行设备所需的操作，最后恢复所保存的状态并执行中断返回指令。
  - **CPU**：中断返回指令恢复**PC**和其他自动保存的状态并返回被中断的代码处继续执行。

Honeywell Internal Use Only
Xi'an Jiaotong University

中断开销

- 中断处理程序的执行时间
- 中断机制相关的执行时间
- 应答中断并获取中断向量的时间
- 中断造成的分支延迟
- 保存和恢复寄存器状态
- 中断返回造成的分支延迟
- 中断造成**流水线和高速缓存**的变化，可能导致延迟

Hongbin Sun
Internal Teaching Use Only
Xi'an Jiaotong University

ARM的中断（1）

- ARM7支持两种类型的中断：
  - 快速中断请求（FIQ）:FIQ优先级高于IRQ
  - 中断请求（IRQ）
- 中断向量表通常保存在内存低部地址，从0地址开始。
- ARM7响应中断的步骤：
  - 保存相应的PC值以便返回；
  - 将CPSR值复制到备份程序状态寄存器（SPSR）；
  - 强制CPSR中的位记录下中断
  - 强制PC指向相应的中断向量

Honeywell Sensing Use Only
Internal Teaching Use Only
Xi'an Jiaotong University

ARM的中断（2）

- 中断处理程序结束前：
  - 用SPSR恢复CPSR
  - 清除中断禁用标志
  - 恢复PC的正确值
- 响应中断的最坏情况（worst case）延迟：
  - 两个周期来同步外部请求
  - 最多20个周期来完成当前指令
  - 三个周期让数据异常终止
  - 两个周期来进入中断处理状态
- 最坏情况总共27个时钟周期，最好情况4个时钟周期。

Hongyun Sun
Internal Teaching Use Only
Xi'an Jiaotong University

管态（supervisor mode）

- 复杂的系统经常由几个相互通信的程序来实现。我们希望能够提供硬件检查来确保程序之间没有互相干扰。
- 可以使用CPU提供的管态来避免内存误操作。MMU控制通常留给管态使用。
- 很多DSP都没有管态（通常也没有MMU）。
- ARM中使CPU进入管态的指令：SWI CODE_1
  - 设置PC为0x08;
  - SWI参数是一个传给管态程序的24位立即数；
  - 执行SWI之前，CPSR的内容保存到保存程序状态寄存器（saved program status register, SPSR）

Hongshen Sun
Internal Teaching Use Only
Xi'an Jiaotong University

异常（exception）

- 异常是一种内部可以检测到的错误。（例：零作除数）
- 通常，可以将异常当做中断的一种变体。
- 一般情况下异常通常需要定义优先级和定向。
  - 异常必须有优先级，因为一个操作可能产生不只一个异常。异常的优先级一般由**CPU**体系结构决定。
  - 定向为用户指定异常状态处理程序提供了一个方法。异常的向量号通常由体系结构预先定义，被用来索引异常处理程序表。

Hongshin Software
International Academic Use Only
Xi'an Jiaotong University

陷阱（trap）

- 陷阱（**trap**），已称为软件中断（**software interrupt**），是一种显式产生异常状态的指令。
- 陷阱最通常的用法是进入管态。
- **ARM**为软件中断提供了**SWI**指令。这个指令将使**CPU**进入管态。一个能被处理程序读取的操作码嵌入到这条指令中。

Hongbin Sun, Internal Teaching Use Only, Xi'an Jiaotong University

协处理器（co-processor）

- 协处理器：由指令来调用的附加功能单元。
- 为了支持协处理器，必须在指令系统中为协处理器操作保留某些操作码。
- 当CPU在没有协处理器的情况下收到协处理器命令：非法指令陷阱，在陷阱程序中在主CPU用软件执行。
- ARM体系结构最多支持16个协处理器，其中编号1和2为浮点协处理器。
  - 浮点协处理器提供了8个80位浮点数据寄存器、浮点状态寄存器。

Honeywell Software Internal Training Use Only
Xi'an Jiaotong University

CPU的功耗

- 能量 vs. 功率
  - 功率是指单元时间内的能量消耗，功耗决定产生的热量
  - 能量消耗在很大程度上决定着电池的寿命。
  - 一般**功耗**作为**能量消耗**和**功率消耗**的简称，只有必要时才区分。
- **CMOS VLSI**功耗的计算公式：

$$P_{total} = P_{dynamic} + P_{static} \qquad P_{dynamic} = \lambda C V_{DD}^2 f$$

λ电平变化比率，f频率，V_{DD}供电电压，C负载电容

- 动态功耗：电压降和切换（频率和有效切换）
- 静态功耗：漏电

Hongjun Sun
International Teaching Use Only
Xi'an Jiaotong University

CPU节省功耗的方法

- 降低供电电压
- 以低频率运行（降低功率损耗而不是能量损耗）
- 通过控制线禁用某些当前执行不需要的功能单元

- 允许部分CPU完全与电源断开以消除泄露电流。

Honeyun Suo Teaching Use Only
Interdisciplinary
Xi'an Jiaotong University