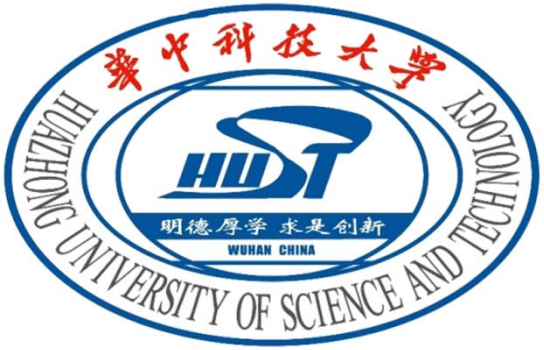
微机原理与接口技术

MicroBlaze MB-GCC C语言中断控制API函数

华中科技大学　左冬红



中断服务程序申明

```
void function_name() __attribute__((interrupt_handler));
// 普通中断模式函数
```

```
void interrupt_handler_name() __attribute__((fast_interrupt));
//快速中断模式函数
```

共同点：

- 都不能带参数
- 编译器编译时函数最后插入中断返回指令

不同点：

- interrupt_handler整个系统只能有一个函数采用此申明，由编译器将该函数地址填写进MB的中断向量表
- fast_interrupt可以多个函数采用此类型，函数地址需由用户程序填入INTC硬件维护的中断向量表

StandAlone BSP API函数

注册中断事务处理函数

```
void microblaze_register_handler(XInterruptHandler Handler, void *DataPtr);
```

函数体

```
MB_InterruptVectorTable[0].Handler = Handler;
MB_InterruptVectorTable[0].CallBackRef = DataPtr;
```

```
MB_InterruptVectorTableEntry MB_InterruptVectorTable[] =
{
{       XIntc_DeviceInterruptHandler,
        (void*) XPAR_MICROBLAZE_0_AXI_INTC_DEVICE_ID}
};
```

MB中断向量表初始值

```
typedef struct
{
  XInterruptHandler Handler;
  void *CallBackRef;
} MB_InterruptVectorTableEntry;
```

MB中断向量表结构体

MicroBlaze开、关中断API函数

开中断

函数原型	汇编代码
void microblaze_enable_interrupts(void)	mfs r12, rmsr ori r12, r12, 2 mts rmsr, r12 rtsd r15, 8 or r0, r0, r0

关中断

函数原型
void microblaze_disable_interrupts(void)

小结

- MicroBlaze C语言中断程序设计
  - 注册中断服务程序
    - 编译器
    - 编译器+IO写INTC
    - StandAlone API （中断事务处理函数，非真正中断服务程序）
  - 开、关中断
    - StandAlone API

真正中断服务程序不能带参数

下一讲：快速中断应用