# 一、回顾STM32中断系统

## 1、STM32中断向量表

ARM芯片从0X00000000开始运行，执行指令。在程序开始的地方存放着中断向量表。中断向量表主要功能是描述中断对应的中断服务函数。

对于STM32来说代码最开始的地址存放堆栈栈顶指针。

## 2、中断向量偏移

一般ARM从0X000000000地址开始运行，对于STM32我们设置连接首地址为0X8000000。

如果代码一定要从0X8000000开始运行，那么需要告诉一下soc内核。也就是设置中断向量偏移。设置SCB的VTOR寄存器为新的中断向量表起始地址即可。

## 3、NVIC中断控制器。

NVIC就是中断管理机构。使能和关闭指定的中断、设置中断优先级。

## 4、中断服务函数的编写

中断服务函数就是中断要做的事情。

# 二、Cortex-A7中断系统

## 1、Cortex-A中断向量表

Cortex-A中断向量表有8个中断，其中重点关注IRQ。Cortex-A的中断向量表需要用户自己去定义。

## 2、中断向量偏移

我们的裸机历程都是从0X87800000开始的，因此要设置中断向量偏移。

## 3、GIC中断控制器。

同NVIC一样，GIC用于管理Cortex-A的中断。GIC提供了开关中断，设置中断优先级。

## 4、IMX6U中断号

为了区分不同的中断，引入了中断号。ID0~ID15是给SGI,ID16~ID31是给PPI。剩下的ID32~1019给SPI，也就是按键中断、串口中断。。。。

6ULL支持128个中断。

## 4、中断服务函数的编写

一个是IRQ中断服务函数的编写，另一个就是在IRQ中断服务函数里面去查找并运行的具体的外设中断服务函数，

# 三、GIC中断控制器

# 四、中断实验编写

## 1、编写按键中断例程。

KEY0使用UART1\_CTS这个IO。编写UART1\_CTS的中断代码。

## 2、修改start.S

添加中断向量表，编写复位中断服务函数和IRQ中断服务函数。

编写复位中断服务函数，内容如下：

①、关闭I,D Cache和MMU。

②、设置处理器9中工作模式下对应的SP指针。要使用中断那么必须设置IRQ模式下的SP指针。索性直接设置所有模式下的SP指针。

③、清除bss段。

④、跳到C函数，也就是main函数

## 3、CP15协处理器

MRC: 将 CP15 协处理器中的寄存器数据读到 ARM 寄存器中。

MRC 就是读 CP15 寄存器， MCR 就是写 CP15 寄存器， MCR 指令格式如下：

MCR{cond} p15, <opc1>, <Rt>, <CRn>, <CRm>, <opc2>

MRC p15, 0, r0, c0,c0,0

现在要关闭I,D ache和MMU，打开Cortex-A7参考手册到105页，找到SCTLR寄存器。也就是系统控制寄存器，此寄存器bit0用于打开和关闭MMU，bit1控制对齐，bit2控制D Cache的打开和关闭。Bit11用于控制分支预测。Bit12用于控制I Cache。

**中断向量偏移设置**

将新的中断向量表首地址写入到CP15协处理器的VBAR寄存器。

MCR{cond} p15, <opc1>, <Rt>, <CRn>, <CRm>, <opc2>

MRC p15,0,r0,c12,c0,0 //

MCR p15,0,r0,c12,c0,0

**IRQ中断服务函数**

mrc p15, 4, r1, c15, c0, 0 读取CP15的CBAR寄存器。CBAR寄存器保存了GIC控制器的寄存器组首地址。GIC寄存器组偏移0x1000~0x1fff为GIC的分发器。0x2000~0x3fff为CPU接口端。意味我们可以访问GIC控制器了！

代码中，R1寄存器吧保存着GIC控制器的CPU接口端基地址。读取CPU接口段的GICC\_IAR寄存器的值保存到R0寄存器里面。可以从GICC\_IAR的bit9~0读取中断ID，我们读取中断ID的目的就是为了得到对应的中断处理函数。

system\_irqhandler就是具体的中断处理函数，此函数有一个参数，为GICC\_IAR寄存器的值。

system\_irqhandler处理完具体的中断以后，需要将对应的中断ID值写入到GICC\_EOIR寄存器里面。