上海大学　计算机学院

《计算机组成原理实验》报告十

姓名 胡才郁　　**学号** 20121034

时间 周四 9-11　　**机位** __　　**指导教师** 刘学民

实验名称：中断机制和应用

一、实验目的

1. 学习实验箱感知中断的硬件结构和工作原理。
2. 学习使用中断系统。
3. 学习使用扩展外设。

二、实验原理

程序中断：因“随机性”原因，使一个程序暂停执行，转而执行另一个程序，以处理随机事件，然后再返回原程序继续执行的过程成为“中断”。中断，是指暂停当前执行的程序，执行其它的程序。为了能够恢复当前的任务，需要保存当前任务执行处的断点。当 CPU 采用中断方式实现主机与 I/O 交换信息时，CPU 在每条指令执行阶段结束前，都要发送中断查询信号，以检测是否有某个 I/O 提出中断请求。如果有请求，CPU 则要进入中断响应阶段，又称中断周期。

一般使用堆栈来保存断点，在每一个程序运行时，操作系统都会为这个程序在主存之中开辟一段空间作为此进程的运行堆栈，使用堆栈指针 SP 指向当前运行堆栈的栈顶元素，进栈操作为先修改指针，后存入数据。

中断同子程序调用有共同点：执行另一个程序，然后返回。所以在调用另一个程序（中断服务子程序）时必须保存断点。

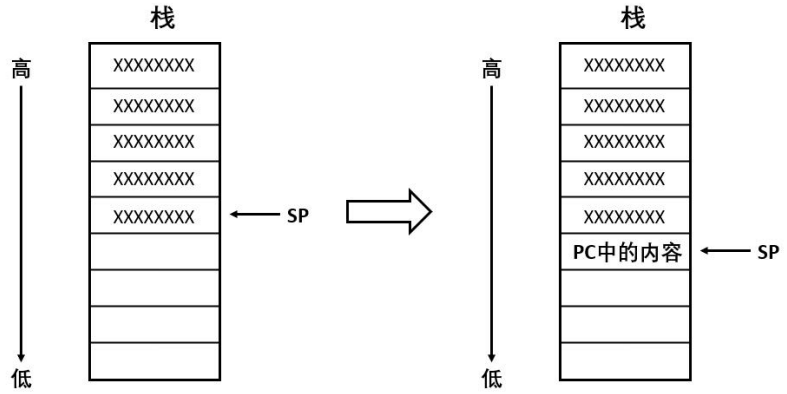

图 1 堆栈寄存器中断点存储示意图

程序中断硬件实现：硬件的逻辑结构如下图。

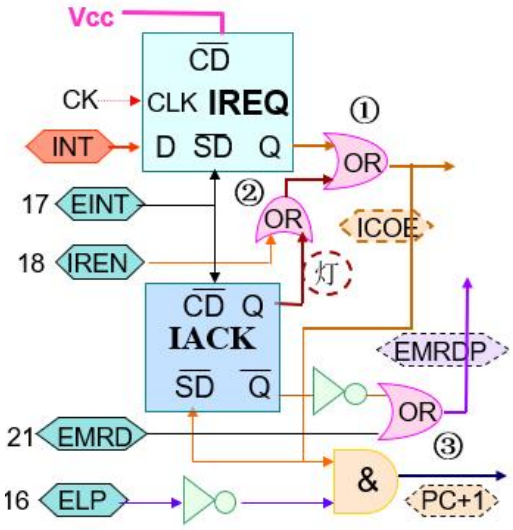

图 2 实验箱的中断感知硬件

只有“中断返回”指令和复位操作使 EINT 为低电平，这个低电平作用到 IREQ 的 SD 端，使上面这个 D 触发器的 Q 端为 1，作用到 IACK 的 CD 端使下面这个 D 触发器的 Q 端输出 0。

系统复位结束或执行其他指令时,EINT 为无效的高电平，这时在时钟 CK 驱动下，IREQ 的 Q 端输出 D 端的 INT 状态。当有中断请求时 INT 为 0，则一个 CK 后 Q 端输出 0，但这个 0 能否被 CPU 感知却要看①号“或门”是否允许它通过。而“非取指”微指令有 IREN＝1，则②号“或门”输出 1，于是 IREQ 的 Q 端无论输出 0 或 1，①号“或门”总输出 1，即不允许中断请求通过。同时这个 1 又送入 IACK 的 SD 端；于是下触发器的 SD 和 CD 端的输入都是无效状态，这个触发器保持稳定。

三、实验内容

1.实验任务一：用 74LS 08 芯片搭建当电键 K1 和 K2 都为 1 时不产生中断请求信号的外部电路。

(1) 实验步骤

①将 74LS08 芯片插入外围芯片实验界面的扩展单元，共使用 5 根导线，其中两根分别对应下图 14、7 端口，接高电平与低电平使得芯片工作， 1A、1B 分别接 K1、K2。1Y 接 INT，即图中黄线红色边框部分。

②分别拨动 K1、K2，给予高电平与低电平信号，记录下 00 01 10 11 四种情况下灯泡的亮灭情况。

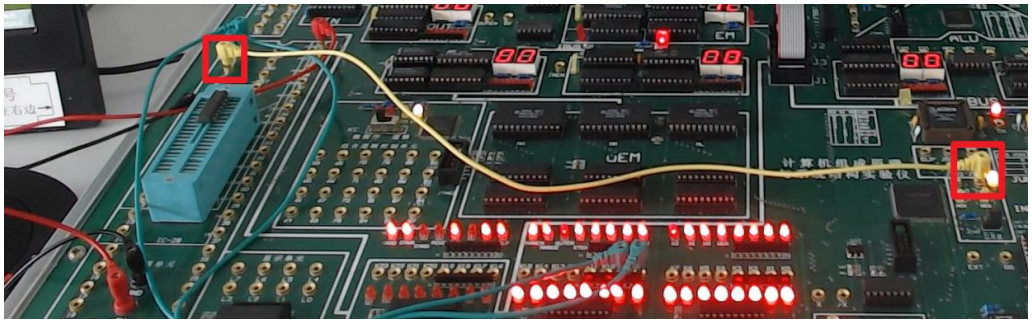

图 3 实验箱的接线

(2) 实验现象

灯泡的情况与 K1、K2 之间的关系如下表所示

表 1 电键与开关情况

K1	K2	灯泡情况
0	0	不亮
0	1	不亮
1	0	不亮
1	**1**	**亮**

(3) 数据记录、分析与处理

根据实验现象可知，当 K1、K2 为 0、0；0、1；1、0 的组合时电路产生中断请求，灯不亮；当 K1、K2 为 1、1 时不产生中断请求信号，灯亮。

由于 74LS08 芯片包含 4 个与门（如下图所示），因此，此处的与门输入输出端除了可以选择为 1A、1B、1Y 之外，也可以选择其余的三个与门，也可以完成实验要求。

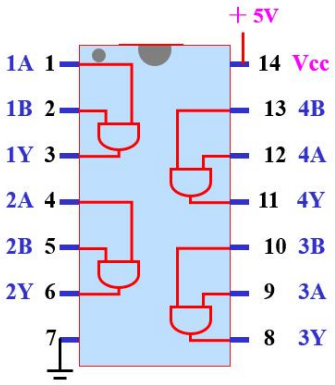

图 4 74LS08 芯片

(4) 实验结论

实现了 74LS 08 芯片搭建当电键 K1 和 K2 都为 1 时不产生中断请求信号的外部电路。

2．实验任务二：编制中断服务子程序使 OUT 交替显示 AA、BB 三次后返回源程序。源程序为上次实验完成的交替显示 11 和 55 的程序。

(1) 运行上述程序，在完成 AA、BB 交替显示三次之前恢复 K1K2 都为 1 的状态。记录 OUT 显示的现象、REQ 灯和 ACK 灯的情况以及 ST 寄存器的值及改变情况。

(2) 运行上述程序，在完成 AA、BB 交替显示时不恢复 K1K2 都为 1 的状态。记录 OUT 显示的现象、REQ 灯和 ACK 灯的情况以及 ST 寄存器的值及改变情况。

分析上述二种显示现象的原因。

(要求：R0：中断时 AA、BB 显示的次数

R1：跟踪主程序 OUT 显示的值

R2：跟踪主程序延时的值）

(1) 实验步骤

①打开集成编程环境。连接实验设备，打开集成编程环境

②连接 PC 机与实验箱的通信口，选择“COM4，打开编写好的后缀为 asm 的程序，编译并下载源程序；

对于 AA、BB 交替显示三次之前恢复 K1K2 都为 1 的状态操作如下：

③置 K1、K2 为 0、0 或 0、1 或 1、0（产生中断请求），此时调用中断程序，

④在完成 AA、BB 交替显示三次之前恢复 K1、K2 都为 1 的状态。

在完成 AA、BB 交替显示时不恢复 K1K2 都为 1 的状态操作如下：

③置 K1、K2 为 0、0 或 0、1 或 1、0（产生中断请求），此时调用中断程序，

④在完成 AA、BB 交替显示三次之后恢复 K1、K2 都为 1 的状态。

现分析该主程序如何实现本实验任务。汇编程序与分析如下表：

表 2 主程序功能分析

	ORG 00H	设置起始地址源为 00H
LOOP:	MOV A, #11H	将立即数 11H 送至累加器 A
	MOV R1, A	累加器 A 中内容送入 R1 寄存器
	OUT	将累加器 A 中内容送至 OUT
	MOV A, #20H	将立即数 20H 送至累加器 A
Z1:	SUB A, #01H	累加器 A 中内容减去 01H
	MOV R2, A	累加器 A 中内容送入 R2 寄存器
	JZ Z2	此处如果遇到零值，跳转 Z2 地址处
	JMP Z1	无条件跳转 Z1 地址处

Z2:	MOV A, #55H	将立即数 55H 送至累加器 A
	MOV R1, A	累加器 A 中内容送入 R1 寄存器
	OUT	将累加器 A 中内容送至 OUT
	MOV A, #20H	将立即数 20H 送至累加器 A
Z3:	SUB A, #01H	累加器 A 中内容减去 01H
	MOV R2, A	累加器 A 中内容送入 R2 寄存器
	JZ Z4	此处如果遇到零值，跳转 Z4 地址处
	JMP Z3	无条件跳转 Z3 地址处
Z4:	JMP LOOP	无条件跳转到 LOOP 地址处

在主程序中，有两个程序主体，即负责交替显示 11H 及 55H 的 LOOP 与 Z2 以及负责延时的程序 Z1 与 Z3。LOOP 与 Z2 程序端功能相似，负责交替显示 11H 及 55H，Z1 与 Z3 功能基本相似，在其中进行递减操作实现延时。

若想要交替实现显示 11H 及 55H，首先将 11H 送入累加器 A 作为每一次循环的开始，OUT 寄存器输出显示 11H 之后，将立即数 20H 送入累加器之后，进入 Z2 部分，在 Z2 部分中输出显示 55H 后，进入 Z3。当由 Z3 跳转到 Z4 时，通过设置 JMP LOOP，再次跳转至主程序开始的入口处，实现了死循环。

现分析中断程序如何实现本实验任务。汇编程序与分析如下表：

表 3 中断程序功能分析

	ORG 60H	设置起始地址源为 60H
	MOV R0, #03H	**将立即数 03H 送至 R0 寄存器**
LOOP1:	MOV A, #AAH	将立即数 AAH 送至 A 寄存器
	OUT	将累加器 A 中内容送至 OUT
	MOV A, #10H;	将立即数 10H 送至 A 寄存器
T1:	SUB A, #01H	累加器 A 中内容减去 01H
	JZ T2	此处如果遇到零值，跳转 T2 地址处
	JMP T1	无条件跳转 T1 地址处
T2:	MOV A, #BBH	将立即数 BBH 送至 A 寄存器
	OUT	将累加器 A 中内容送至 OUT
	MOV A, #10H;	将立即数 10H 送至 A 寄存器
T3:	SUB A, #01H	累加器 A 中内容减去 01H
	JZ T4	此处如果遇到零值，跳转 T4 地址处
	JMP T3	无条件跳转 T3 地址处

T4:	**MOV A, R0**	**R2 寄存器中内容送入累加器 A**
	SUB A, #01H	**累加器 A 中内容减去 01H**
	JZ EXIT	此处如果遇到零值，则退出
	MOV R0, A	累加器 A 中内容送入 R2 寄存器
	JMP LOOP1	无条件跳转 LOOP1 地址处
EXIT:	MOV A, R1	R1 寄存器中内容送入累加器 A
	OUT	将累加器 A 中内容送至 OUT
	MOV A, R2	R2 寄存器中内容送入累加器 A
	SUB A, #00H	累加器 A 中内容减去 00H
	RETI	
	END	

而在中断程序中，交替输出 AA 与 BB 的方式与在主程序之中类似 LOOP1 与 T2 负责交替显示，T1 与 T3 负责延时。而与主程序的差别主要存在于 T4 处，通过在进入 LOOP1 之前在 R0 寄存器之中存入立即数 03H，在 T4 之中对 R0 中的内容递减，来控制中断程序的循环次数。

因此，此中断程序将循环交替显示 AAH 与 BBH 三轮，而主程序为死循环，交替显示 11H 与 55H。

(2) 实验现象

主程序阶段：当启动程序时，此时处于主程序的执行状态，可以发现，OUT 寄存器中，有 11H 与 55H 交替显示。而 R0 显示为 0，R1 中值与 OUT 寄存器同步，R2 记录的是主程序的延迟时间，此时 REQ 与 ACK 灯都不亮。

中断程序阶段：此时拨动中断请求开关，会产生一个中断请求信号，此时两个灯亮起，REQ 为请求信号，ACK 为中断响应信号，此时主机响应中断，OUT 中交替显示 AA 与 BB。R0 记录 AA 和 BB 交替次数，依次出现 03、02、01。R1 保存中断时 OUT 值，视频中为 55H；R2 保存中断时主程序的延迟时间。

对于 AA、BB 交替显示三次之前恢复 K1K2 都为 1 的状态实验现象如下：

AA、BB 交替显示三次之前恢复 K1、K2 都为 1 的状态，将中断信号复位，ACK 灯与 REQ 灯不亮，从 R2 记录的延迟时间开始，OUT 显示 R1 记录的主程序的值（本次实验中为 55），之后交替出现 11 和 55。断点存储在 R2 寄存器之中。当中断程序结束之后，主程序延时运行位置从 R2 中的值继续往下执行。

在完成 AA、BB 交替显示时不恢复 K1K2 都为 1 的状态实验现象如下：

在完成 AA、BB 交替显示三次之后恢复 K1、K2 都为 1 的状态，ACK 灯和 REQ 仍保持亮的状态，OUT 寄存器一直交替显示 AA 与 BB。

(3) 数据记录、分析与处理

分析以上实验现象，当给出中断信号时，主程序的延时位置保存在 R2 寄存器之中，OUT 寄存器中的值保存在 R1 寄存器之中。如果给与中断信号，则执行中断程序。尽管中断程序内循环只有 3 轮，但如果在中断程序结束之后仍不将中断信号复位，则又将发送中断请求，则继续执行中断程序，出现了 AA 与 BB 无限循环的实验现象。

(4) 实验结论

当发送中断请求时，将执行中断程序。此时需要将中断信号复位，否则会出现中断程序“死循环”的现象。

四、建议和体会

本次实验中学校课程为学生准备了丰富的预习资料，因此学生仅需要一步一步从无到有跟随教学内容前进即可完成预习工作与实验任务，本次实验教学效果良好，本人没有建议。本次实验我收获颇丰，体会良多。

经过本次实验，我通过观看视频，了解了中断程序的运行机制，并且记住了在给予中断信号后，要及时中断信号复位，否则会出现中断程序死循环的意外情况。

五、思考题

问题：实验箱的中断服务程序中可以嵌套一般的子程序吗？

答：不可以。

子程序嵌套是指可以让子程序调用另一个子程序，然而，在调用子程序时需要保存当前程序执行位置的断点，而对于中断服务程序而言，其本身已经占用了一个断点位置，而且本实验箱的 ST 堆栈寄存器只有一层，只能存储一个断点地址，所以不能嵌套一般的子程序。