

北京交通大学汇编与接口技术实验报告

姓名：程维森

学号：21231264

实验四、中断方式的A/D采集系统

* 1. 实验目的

该实验使用了8259A，ADC0809，数码管来完成一个数据采集系统的设计，目的是了解中断方式的A/D采集数据的实现方法，掌握硬件设计和中断程序的编写方法，是对学生综合实验能力的训练。

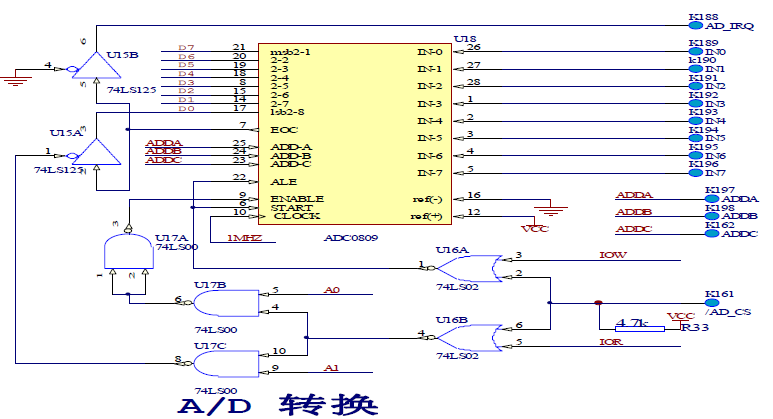
* 1. 实验内容

使用ADC0809的通道0，接入0-5V的直流电压，用WR电位器调整模拟电压值，A/D的转换结束信号EOC接在主8259A的IR5上，采集100个数据并存入内存中，同时将采集的16进制数据显示在数码管上。请多次调整0-5V的电压值（旋动W1旋钮），进行A/D采集，并观测内存中的数据的变化情况。

* 1. 实验现象

每次采集的100个数据可能是相同的（数码管的数据也可能不变），当WR旋动时可以采到不同的数据。

* 1. 实验接线图



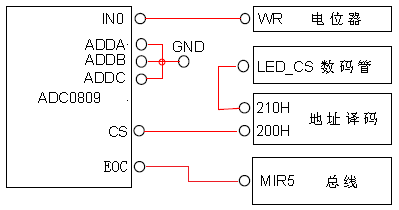


图1-6 实验连线图

* 1. 实验编程提示

主程序要对8259A设置中断向量，开中断，检测采样次数，第1次启动A/D的工作要在主程序里做；中断服务程序里要采集数据、存储数据、并启动下一次A/D转换。采集到的数据宽度是8位，要用两个数码管来分别显示高4位和低4位的值。

* 1. 实验步骤
     1. 根据原理图正确连接实验线路（需要连接红线）。
     2. 正确理解实验原理。
     3. 编写实验程序，并上机调试，观察实验结果。

*一些说明：*

*ADC0809的特性：其测量的电压范围是0---5V，采样分辨率为8位。WR电位器的电压调节范围也是0-5V。*

*ADC0809的EOC信号：转换结束信号，在实验箱上对应的是AD\_IRQ信号。EOC=0，正在进行转换；EOC=1，转换结束。*

*ADC0809的ENABLE信号：相当于课件中的OE信号，都是ADC的数据输出允许信号，只不过叫法不同。ENABLE=0，输出数据线呈高阻；ENABLE =1，输出转换得到的数据。*

*ADC0809的START信号的控制：START信号是正脉冲有效，初始化时，ST（是START的简称）设为低电平。课件PPT上电路原理图中START接的是电平信号，所以要先写高电平，再写低电平，才能获得一个正脉冲。而实验课所用的实验箱中，根据实验原理图，其接的是OUT/IN指令产生的脉冲信号，其控制过程就不需要先写高电平再写低电平，但是需要仔细设计OUT/IN指令的操作数。*

*ADC0809的数据端口地址是怎么安排的？*

*答：根据PDF实验指导书的电路原理图，*

*1，读数据过程中用到的某奇数端口地址：ENABLE信号和OE信号是一样的，只不过叫法不同。 ENABLE信号的有效，取决于三根控制线/AD\_CS、/IOR和A0。当这三根线分别为0，0和1时，即执行“IN AL，奇数端口地址”指令，使得ENABLE信号有效，此时ADC0809上输出数据到CPU的AL寄存器。只要是奇数端口地址即可，不关心具体什么样的奇数端口地址。*

*2，启动ADC转换过程中用到的某偶数端口地址：ADC0809的启动转换过程是靠/IOW和/AD\_CS的来控制的，只要执行OUT指令的操作码即可让/IOW和/AD\_CS都低电平。在考虑OUT指令的操作数的过程时，任意的数据都可以，但是在端口地址上，需要注意不要让A0为1 （因为不太清楚/IOW和/IOR是否是互斥的。如果两者不是互斥的，那么如果将A0设为1，那么在启动转换过程的同时，也会启动读数据的过程，就会有错误发生）。因此，为保险起见，还是将OUT指令的端口地址设为偶数地址。*

# 三．实验内容

实验内容

电路连接和设置

使用ADC0809的通道0，接入0-5V的直流电压。

通过WR电位器调整模拟电压值。

EOC（转换结束信号）连接在主8259A的IR5上。

采集过程

在主程序中设置中断向量、开启中断，以及检测采样次数。

第一次启动A/D的工作在主程序中进行。

中断服务程序中进行数据采集、存储数据，并启动下一次A/D转换。

数据显示

采集到的数据为8位，需使用两个数码管分别显示高4位和低4位的值。

实验现象

每次采集的100个数据可能相同（数码管数据也可能不变）。

通过旋动WR旋钮可采集到不同的数据。

实验步骤

连接电路

根据实验连线图正确连接电路（注意红线连接）。

理解原理

理解实验原理和各器件的工作原理。

编写程序

在主程序中设置中断向量，开启中断，并监测采样次数。

中断服务程序中实现数据采集、存储和启动下一次A/D转换。

使用两个数码管分别显示采集数据的高4位和低4位。

电路组成要点

ADC0809特性

采样电压范围：0-5V，分辨率为8位。

EOC信号：转换结束，连接在AD\_IRQ信号上。

ENABLE信号：ADC数据输出允许信号。

START信号：正脉冲有效。

ADC0809连接端口地址

读数据过程中使用的奇数端口地址：通过IN指令将输出数据读入CPU的寄存器。

启动ADC转换过程中使用的偶数端口地址：通过OUT指令启动转换过程。

# 实验代码分析

实验代码是为了实现一个数据采集与输出的功能，以下是实验的代码综述：  
实现功能：

系统设置

定义了堆栈段 STACKS 和数据段 DATA。

初始化寄存器和设备接口，包括中断向量的设置。

ADC0809 数据采集

使用 ADC0809 模数转换器，通过端口 IO0809A 接口读取模拟信号。

主程序中设置了 A/D 转换工作，并在中断服务程序中进行数据采集和存储。

通过中断控制器 8259A 的 IR5 连接转换结束信号 EOC，在中断服务程序中启动下一次 A/D 转换。

数码管显示

采集到的 8 位数据经过处理后，通过两个子程序 DISP1 和 DISP2 在数码管上显示。

DISP1 和 DISP2 分别控制数码管显示的高 4 位和低 4 位数据。

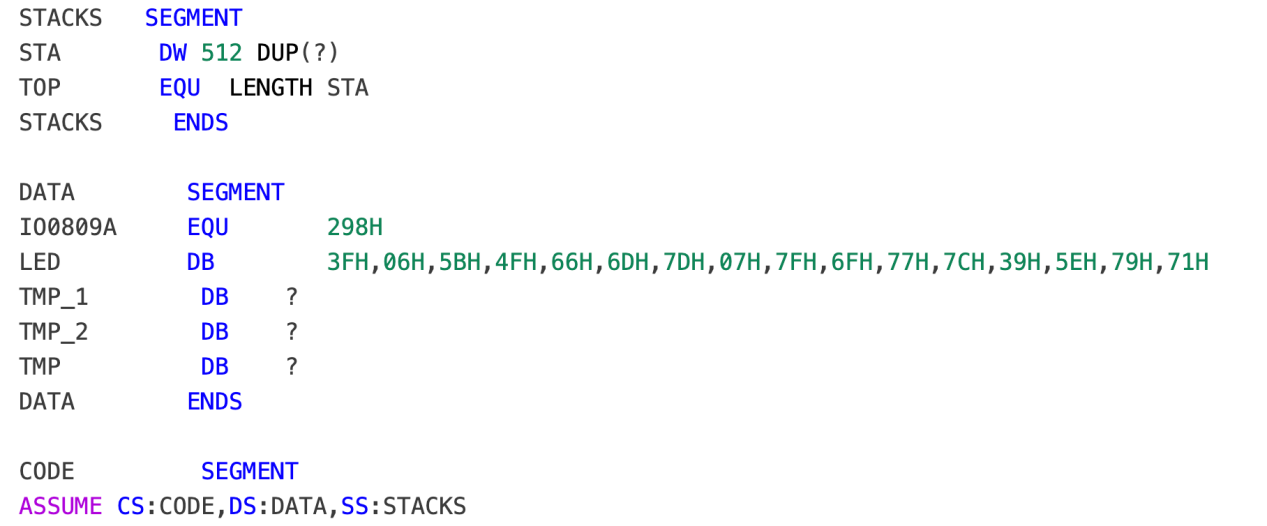
其他功能

对中断进行了管理和处理，包括中断的开启和关闭。

使用了堆栈来保存和恢复寄存器状态。

通过输入/输出指令进行数据的读取和输出。

下面是代码的分区说明：



具体描述如下：

段定义：

STACKS SEGMENT 定义了一个名为 STACKS 的段。

STA DW 512 DUP(?)：在 STA 中定义了一个 512 个字节大小的堆栈。

TOP EQU LENGTH STA：TOP 被赋值为 STA 的长度。

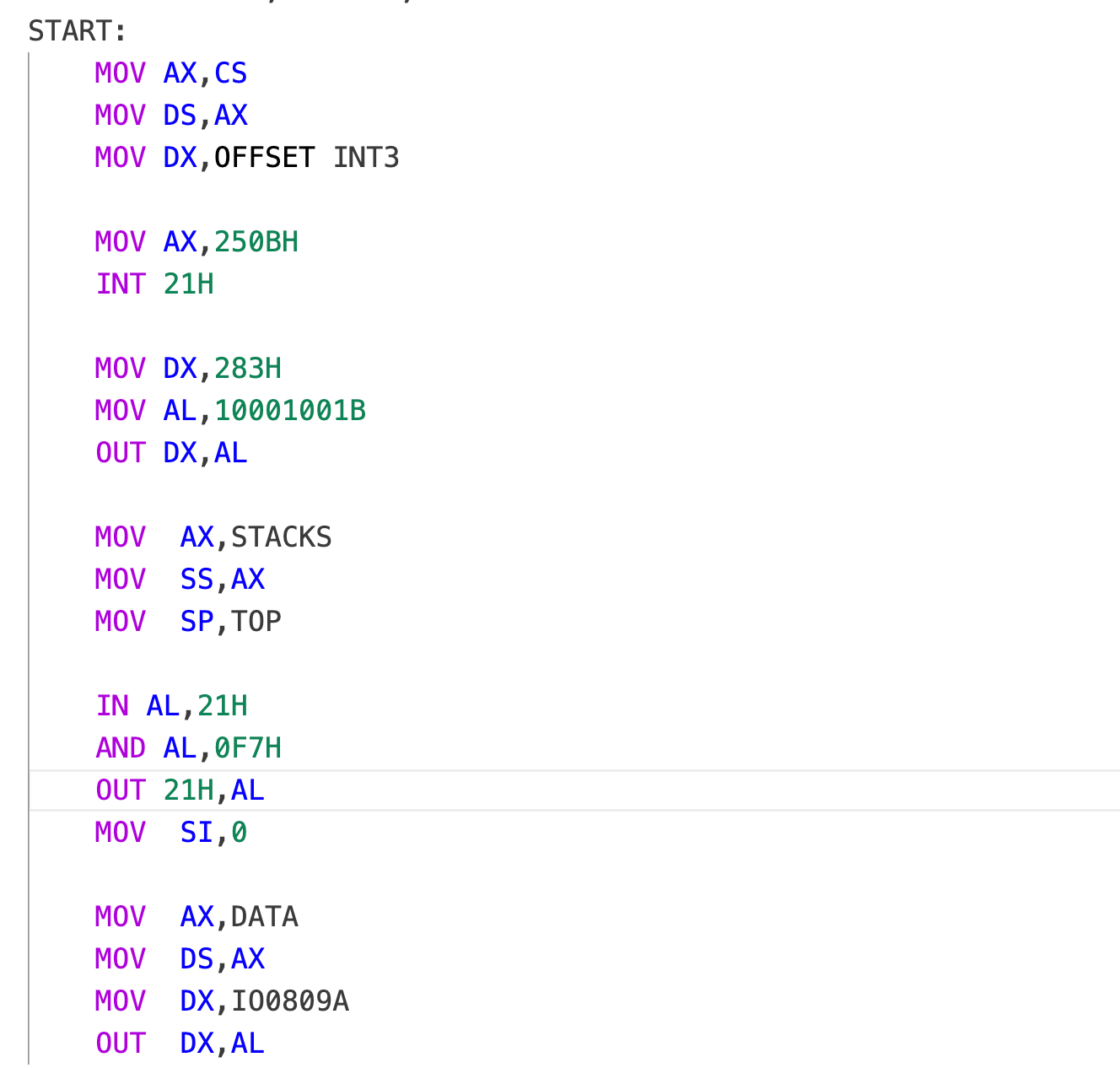
DATA SEGMENT 定义了一个名为 DATA 的段。

IO0809A EQU 298H：IO0809A 被赋值为十六进制地址 298H，可能表示设备的接口或端口地址。

LED DB ...：LED 是一个数据段，包含了一系列十六进制数据。

段指定及假设：

ASSUME CS:CODE, DS:DATA, SS:STACKS：这里进行了段寄存器的假设，指定了代码段 CODE、数据段 DATA、以及堆栈段 STACKS 分别与 CS、DS 和 SS 寄存器相对应。



这段汇编代码的主要部分包括以下操作：

设置段寄存器和中断向量：

MOV AX, CS：将代码段寄存器 CS 的内容加载到通用寄存器 AX 中。

MOV DS, AX：将通用寄存器 AX 的内容传送到数据段寄存器 DS，用于设置数据段。

中断调用：

MOV DX, OFFSET INT3：将中断服务程序 INT3 的偏移地址加载到数据寄存器 DX。

MOV AX, 250BH：将中断号 250BH（在此情景下可能表示 A/D 转换结束的中断号）加载到寄存器 AX。

INT 21H：通过中断 21H 调用启动中断服务程序，可能是为了初始化或配置某些功能。

设备端口操作：

MOV DX, 283H：将地址 283H 装载到数据寄存器 DX，通常用于对设备端口进行操作。

MOV AL, 10001001B：将二进制值 10001001B 装载到累加寄存器 AL 中。

OUT DX, AL：将累加寄存器 AL 中的数据输出到地址为 DX 的设备端口上，可能是对某个设备进行设置或控制。

堆栈和数据操作：

MOV AX, STACKS：将堆栈段的起始地址加载到通用寄存器 AX。

MOV SS, AX：将通用寄存器 AX 的内容传送到堆栈段寄存器 SS，用于设置堆栈段。

MOV SP, TOP：将堆栈段顶部地址 TOP 的值加载到堆栈指针寄存器 SP 中。

端口输入输出操作：

IN AL, 21H：从地址为 21H 的设备端口读取数据到累加寄存器 AL 中。

AND AL, 0F7H：将累加寄存器 AL 中的数据和十六进制值 0F7H 进行按位与操作。

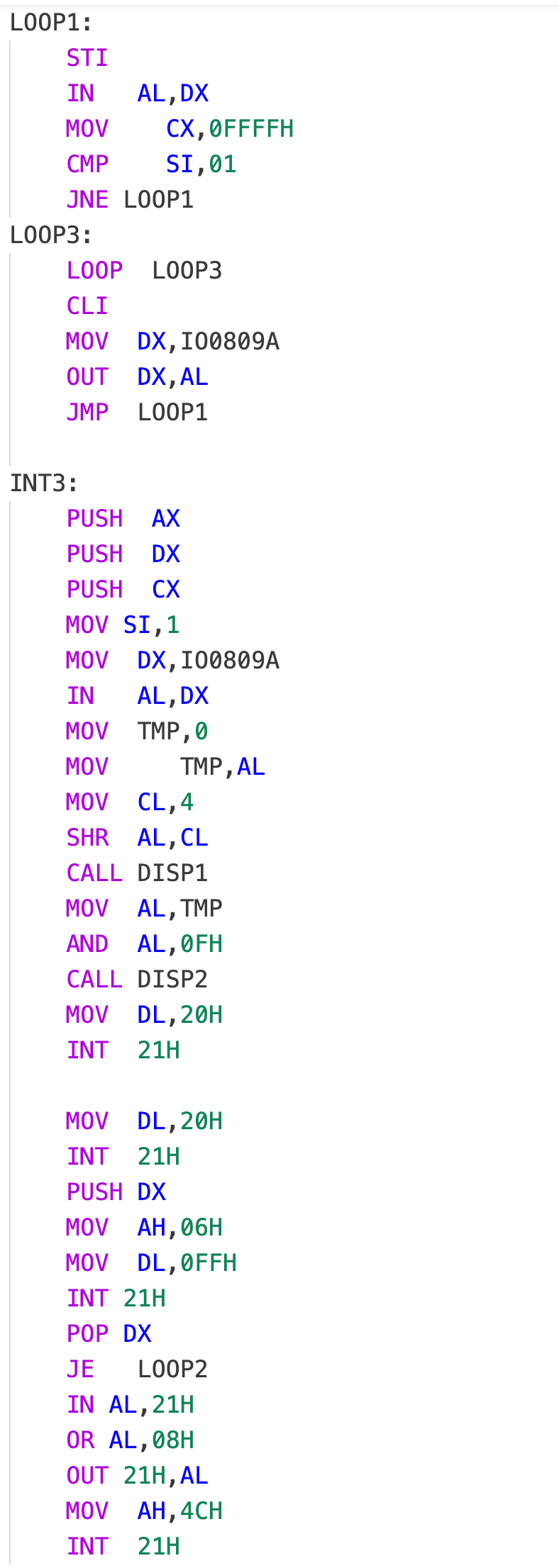
OUT 21H, AL：将累加寄存器 AL 中的数据输出到地址为 21H 的设备端口上。

数据段操作：

MOV AX, DATA：将数据段的起始地址加载到通用寄存器 AX。

MOV DS, AX：将通用寄存器 AX 的内容传送到数据段寄存器 DS，用于设置数据段。

MOV DX, IO0809A：将地址 IO0809A 装载到数据寄存器 DX，可能用于指定某种设备或接口。



主循环部分（LOOP1 和 LOOP3）：

LOOP1：包含了两个嵌套循环。

STI：设置中断允许标志。

IN AL, DX：从设备端口 DX 中读取数据到累加寄存器 AL。

MOV CX, 0FFFFH：将循环计数器 CX 设置为 65535。

CMP SI, 01：将 SI 寄存器与 1 进行比较。

JNE LOOP1：如果 SI 不等于 1，则跳转回 LOOP1 继续执行。

LOOP3：内部是一个简单的循环。

LOOP LOOP3：根据 CX 寄存器的值进行循环（循环 65535 次）。

中断处理部分（INT3）：

INT3：可能是中断服务例程。它执行以下操作：

保存现场：将寄存器 AX、DX 和 CX 的值依次压入堆栈。

MOV SI, 1：将 1 存入寄存器 SI。

MOV DX, IO0809A：将设备端口地址加载到寄存器 DX。

IN AL, DX：从设备端口 DX 中读取数据到累加寄存器 AL。

处理数据：

将 AL 的值保存到 TMP。

右移 AL 的值 4 位并调用 DISP1 进行数码管高 4 位的显示。

对 TMP 取低 4 位并调用 DISP2 进行数码管低 4 位的显示。

中断服务程序结束，进行一些操作：

在屏幕上显示空格。

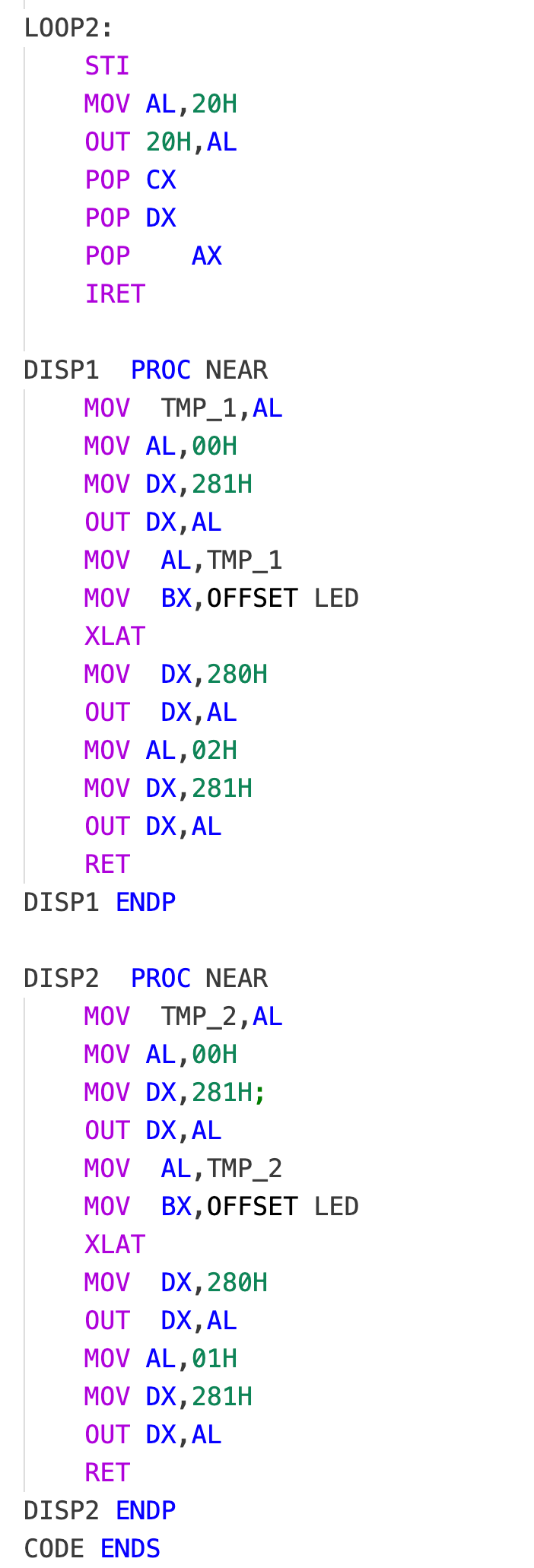
进行键盘输入操作。

根据条件进行输出操作或跳转到 LOOP2 进行下一步处理。

整体功能描述：

主循环部分 LOOP1 和 LOOP3 负责从设备端口读取数据，通过 IN 指令循环地读取。

中断服务程序 INT3 处理了从设备端口读取的数据，将高低 4 位显示在数码管上，并进行一些交互操作，例如在屏幕上显示字符，并根据条件进行输出或跳转。



LOOP2：

STI：设置中断允许标志，允许中断。

MOV AL, 20H：将十六进制值 20H（32）加载到累加寄存器 AL 中。

OUT 20H, AL：向端口地址 20H 输出累加寄存器 AL 中的值。

POP CX：将堆栈中保存的 CX 寄存器的值弹出。

POP DX：将堆栈中保存的 DX 寄存器的值弹出。

POP AX：将堆栈中保存的 AX 寄存器的值弹出。

IRET：中断返回指令，将 CPU 控制权还给被中断的程序。

DISP1 和 DISP2（子程序）：

DISP1：处理数码管的显示。

TMP\_1 是一个暂存变量，存储 AL 寄存器中的数据。

OUT DX, AL：向端口地址 DX 输出累加寄存器 AL 中的数据。

BX 装载了 LED 的偏移地址，XLAT 指令用于根据 AL 的值获取 LED 数据段中相应位置的数据。

MOV AL, 02H：将 AL 寄存器设置为 02H。

OUT DX, AL：向端口地址 DX 输出累加寄存器 AL 中的值，用于控制数码管显示的动作。

RET：返回指令，返回到调用它的地方。

DISP2：类似于 DISP1，用于处理另一组数码管的显示，可能是显示不同的数据或动作。

# 实验现象说明

