专业：电气工程及其自动化
姓名：严旭铧
学号：3220101731
地点：紫金港东三 406

浙江大学 实验报告

课程名称：微机原理与应用综合实验 指导老师：胡斯登 成绩：

实验名称：BCD 控制 实验类型：微机实验 同组学生姓名：褚玘铖

实验 2　BCD 控制

一、 题目要求

1. 30H 与 31H 中存放 4 位 BCD 码数字，编写程序将数字倒序排列
2. 在 RAM 31H 单元存放一组 8 位带符号数，字节个数放在 30H 中，请编写程序统计出其中正数，负数以及 0 的数目，结果存放在 41H，42H 以及 43H 中。
3. 模拟下列逻辑运算编写程序并将运算结果转换为显示码后进行显示。设:A=63H、B=82H、C=0C5H、D=36H

$$Y = \overline{A \oplus B \bullet \overline{C} \bullet \overline{D + A}}$$

二、 代码及结果实现

#注：代码用了截图和 OpenDocument Text 对象插入，后者双击即可编辑。其中注释部分有加 # 的部分是值得注意的地方。

1. EX1

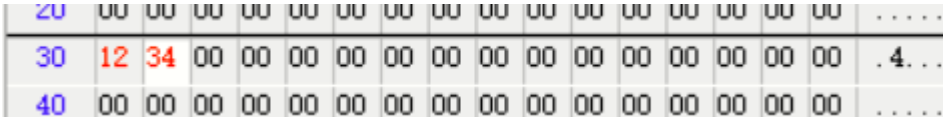
a) 代码

```
;##两步：1. 各数倒序
;2. 两数交换
LJMP    BEGIN
ORG     0030h
BEGIN:
MOV R0,#30H
MOV R1,#31H
MOV A,@R0
SWAP A ;交换 A 高低位，12变21
XCH A,@R0;注意 XCH 对 A 操作，因此这里直接将本来放回@R1再与@R0交换的数
;放到@R0。
MOV @R1,A ;将原来@R0中倒序后的数放入@R1
HERE:SJMP HERE
```

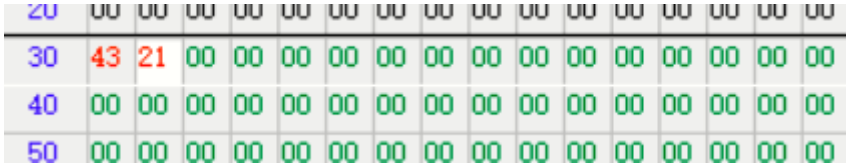
- 思路：
- 先各数倒序，再两数交换。利用 XCH 指令完成两个地址之间的内容互换，再利用 SWAP 对每个地址里面的 2 为 BCD 码进行互换。
- 需要注意，XCH 和 SWAP 都是对 A 进行操作的指令，因此需要用 A 作为中介，将两个寄存器的内容互换，并且处理高低位。

b) 结果

输入：30H 输入 12,31H 输入 34，预期输出为 43 21



输出：



达到实验预期效果

2. EX2

a) 代码

```
BEGIN:
MOV 41H,#0 ;初始化正数计数器
MOV 42H,#0 ;初始化负数计数器
MOV 43H,#0 ;初始化0计数器

MOV  R1,30H;数据长度，用于控制循环次数
MOV  R0,#31H

LOOP: ;循环主体
MOV  A,@R0
JB   ACC.7,CNT_NEGA;##本题核心指令，ACC.7是 A 中数的最高位，存放带符号数
;最高位就是符号位，若为1则为负数，跳转到负数计数器
;否则顺序进行，起到 if-elif-else 的作用，下一条类似
JZ CNT_ZERO;#判断0，为0跳转，否则就剩正数
;# 此 处 也 可 采 用  CJNE A,#0,CNT_POSI,正数跳转，对应着下一条指令要改成 INC 43H
INC  41H;正数计数器+1
JMP LP;这条可不要，保留是为了与另两个分支判断格式一致，好看
LP:  INC R0;##此处是与 C 的不同，下面类似调用函数，C 是调用，汇编是跳转
;跳转了还得回来，因此配对下面的计数“函数”，相当于回城点。
DJNZ R1,LOOP ;控制循环次数

LJMP STOP;如果这里直接用 END，会报错；如果在最后用 END
;由于顺序执行，会一直跑计数器加1并且去 LP，R1会减到0FF 再减，出问题
;因此这里设一个传送点，与下面“函数”定义区隔开
;下面类比为函数定义区
CNT_POSI:INC 41H；正数
AJMP LP;“调用”完了跳转回去，回城
CNT_NEGA:INC 42H；负数
AJMP LP
CNT_ZERO:INC 43H；0
AJMP LP

STOP:SJMP STOP;正式的结束位置
```

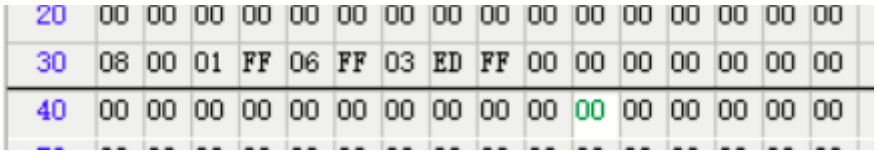
- 思路：
- 正负判断：利用 JB ACC.7,rel 这条指令进行负数的判断。
- 0 的判断：利用 CJNE A,#0 判断;也可使用 JZ rel 判断。两种方法的逻辑是相反的。这两条指令是最核心的指令。

  这里在写的时候没有使用 RET。可以使用 ACALL/LCALL+RET 进行子

程序的调用和返回，可以用 RET 结尾来实现跳转回去。

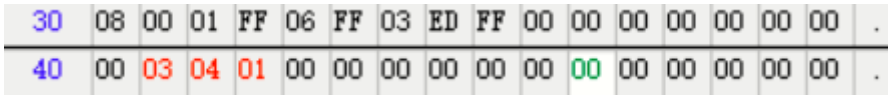
b) 结果

输入：8 个带符号数，1 个 0,3 个正数，4 个负数，41H~43H 应该分别为 03,04,01



20	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
30	08	00	01	FF	06	FF	03	ED	FF	00	00	00	00	00	00
40	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00

输出：



30	08	00	01	FF	06	FF	03	ED	FF	00	00	00	00	00	00
40	00	03	04	01	00	00	00	00	00	00	00	00	00	00	00

得到所求结果

3. EX3

a) 代码

```
BEGIN:
; 初始化寄存器
MOV 31H,#63H
MOV 32H,#82H
MOV 33H,#0C5H
MOV 34H,#36H

MOV R0, 31H
MOV A,R0
XRL 32H,A ； A XOR B,结果放在32H 中

; 计算 NOT C
MOV A,33H
CPL A
ANL 32H,A;
MOV A,32H
CPL A
MOV 32H,A;前面一部分算完放在 32H 备用

MOV A,34H
ORL A,31H
CPL A ;算完或非
ANL A,32H ;最后与，直接留在 A 里
MOV 40H,A ;以16进制呈现在40H 单元
;逐位显示部分 ，实现了将40H 单元字节逐位依次存放在41H~48H 单元中
MOV R0,#41H;起始地址放入 R0
MOV R1,#8 ;一个字节，8位 ，控制循环次数
LOOP:
RLC A ;##带位左移，这样高位先呈现，符合从左到右的顺序 ,并且最高位放入 CY 中
JC PUT1 ;如果 A 的最高位为1，那么这里会直接跳转到 PUT1
MOV @R0,#0 ;相当于 else，最高位是0就在 R0存放的地址中放0
LP:INC R0 ;##这里借鉴 EX2的想法是一个给 PUT1的传送点，指令本身实现“逐位”
DJNZ R1,LOOP ;控制循环次数
HERE:SJMP HERE ;结束指令，和“函数”定义区隔开

PUT1: ;用来放1
MOV @R0,#1
AJMP LP ;回城，不然会乱跑
```

- 思路：
- 逻辑运算题需要考虑逻辑运算顺序，一般是从里到外，最后化为

A？Y 的形式，进行最后一步的逻辑运算。逻辑运算不难，慢慢拆分组合就行。

- 本题还要求用显示码表示结果，但是这个显示码的意思我不是很清楚。我自己理解成给他可视化放入地址中。由于是逻辑运算，结果必然是 8 位的 0 或 1。这里我选择把最后储存在 A 里面的结果（88H）逐位表示出来，在 41H 到 48H 中显示。这样每一位的逻辑取值会很清晰。这一步的核心是 RLC A＋JC PUT1 指令，实现了逐位的 01 判断。也可以改成 JNC，那就是反一下，CY 为 0 跳转

b) 结果

输入部分直接在代码了，也可以改成自己输入值再存进去

输出：

30	00	63	DF	C5	36	00	00	00	00	00	00	00	00	00	00
40	88	01	00	00	00	01	00	00	00	00	00	00	00	00	00

如果没算错的话最后的结果是（1000 1000）B，也就是 88H

三、 心得体会

BCD 码显示可以将不太熟悉的 16 进制转换成十进制的形式显示出来，非常方便。这次三道题的核心我觉得是要熟练运用 A 和 CY 位，因为都是涉及到位的判断和处理的。JC 等对位判断跳转的指令在这类场景下非常有用。