专业：电子科学与技术
姓名：
学号：
日期：
地点：玉泉

浙江大学实验报告

课程名称：微机原理与接口技术 指导老师：____ 成绩：_____
实验名称：第二次软件实验 实验类型：软件实验 同组学生姓名：无

实验三

1、实验目的

掌握算术运算类、逻辑运算类指令的使用方法；掌握 BCD 码、补码数制表示方法；掌握运算程序及循环程序的编写和调试方法。

2、基础实验部分

（1）完成单字节的 BCD 码加法功能，补全程序如 Code1 所示。实验结果如图 1~图 2 所示。

```
RESULT EQU 30H
ORG 0000H
MOV A, #99H
MOV B, #99H
ADD A, B
DA A                    ;BCD 码相加得到结果
MOV RESULT, A
MOV A, #00H
ADDC A, #00H
MOV RESULT+1, A         ;高位处理
SJMP $
END
```

Code 1 实验三基础实验 1 程序

图 1 实验三基础实验 1 片内 RAM 数据

名称	值	名称	值
R0	00	CY	0
R1	00	AC	0
R2	00	F0	0
R3	00	RS1	0
R4	00	RS0	0
R5	00	OV	0
R6	00	F1	0
R7	00	P	1
A	01		
B	99		
DPH	00		
DPL	00		
PSW	01		
SP	07		

图 2 实验三基础实验 1 寄存器数据

注意到程序运行结束后，片内 RAM 地址 30H、31H 存储的数据结果为 $198_{BCD}=99_{BCD}+99_{BCD}$，故运行结果正确。

（2）完成多字节 BCD 码加法运算。内部 RAM 30H 开始的 4 字节长的 BCD 码和外部 RAM 1000H 开始的 4 字节长的 BCD 码相加，结果放在 1100H 开始的单元中（从低字节到高字节）。实验结果如图 3~图 4 所示。

10C0	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF
10D0	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF
10E0	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF
10F0	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF
1100	30	64	33	18	01	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF
1110	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF

地址: 1104H

图 3 实验三基础实验 2 片外 RAM 数据

名称	值	名称	值
R0	34	CY	1
R1	10	AC	1
R2	04	F0	0
R3	11	RS1	0
R4	04	RS0	0
R5	00	OV	0
R6	00	F1	0
R7	00	P	1
A	01		
B	00		
DPH	11		
DPL	04		
PSW	C1		
SP	07		

图 4 实验三基础实验 2 寄存器数据

本次实验选择的两个 BCD 加数分别为 19191919_{BCD} 和 99144511_{BCD}，其相加的结果为 118336430_{BCD}，从低字节到高字节分别与片外 RAM 地址 1100H~1104H 中的数据相对应，故运行结果正确。

3、设计实验部分

（1）实现多字节十六进制数的减法 123456H－005634H，使用单步、断点方式调试程序。程序如 Code2 所示，断点设置在 L1 处。每次 L1 循环完毕及程序运行完毕后，内存单元和寄存器的数据如图 5~图 6 所示。

```
        ORG 0000H
        MOV R3, #03H
        MOV R0, #30H
        MOV R1, #40H
        MOV R2, #50H
        CLR C
L1:     MOV A, @R0      ;取出源数据
        SUBB A, @R1
        MOV B, R1       ;保护 R1
        MOV R1, 02H
        MOV @R1, A      ;保存数据
        INC R2          ;改变目标地址
        MOV R1, B       ;恢复 R1
        INC R1          ;改变源数据地址
        INC R0
L2:     DJNZ R3, L1     ;循环
        JNZ L3
        MOV R1, #53H
        MOV @R1,#0FFH
L3:     NOP
        SJMP $
        END
```

Code 2 实验三设计实验 2 程序

名称	值	名称	值
R0	31	.7	0
R1	41	.6	0
R2	51	.5	1
R3	02	.4	1
R4	00	.3	0
R5	00	.2	0
R6	00	.1	0
R7	00	.0	1
A	22		
B	40		
DPH	00		
DPL	00		
PSW	00		
SP	07		

名称	值	名称	值
R0	32	CY	1
R1	42	AC	1
R2	52	F0	0
R3	01	RS1	0
R4	00	RS0	0
R5	00	OV	0
R6	00	F1	0
R7	00	P	0
A	DE		
B	41		
DPH	00		
DPL	00		
PSW	C0		
SP	07		

名称	值	名称	值
R0	33	CY	0
R1	43	AC	0
R2	53	F0	0
R3	00	RS1	0
R4	00	RS0	0
R5	00	OV	0
R6	00	F1	0
R7	00	P	0
A	11		
B	42		
DPH	00		
DPL	00		
PSW	00		
SP	07		

图 5 实验三设计实验 2 寄存器单步断点调试数据变化

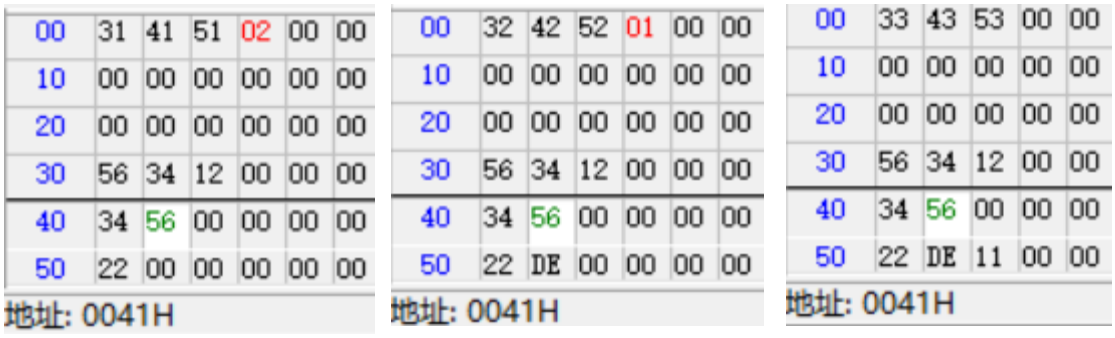


图 6 实验三设计实验 2 片内 RAM 单步断点调试数据变化

注意到在第二次循环时，由于 45H-56H 最高位出现了借位，因此 Cy=1。每次循环结束时寄存器 A 保存了当前字节的减法结果，并保存至相应的 RAM 单元，所以保存结果的片内 RAM 地址 50H~52H 的数据在循环中依次被设为了 22H、DEH、11H，最终的结果为 11DE22H。

（2）在内部 RAM 的 30H 单元开始，有一串带符号数据块，其长度在 10H 单元中。编程求其中正数与负数的和，并分别存入 2CH 与 2EH 开始的 2 个单元中。（负数存放形式为补码）。分别在 30H 单元开始写入 5 个正数、11 个负数和 9 个正数、7 个负数的情况，记录程序运行结果。程序如 Code3 所示。

首先写入数据：-69H、65H、-7H、-25H、-47H、56H、-3H、17H、6H、-5H、-45H、-1H、 -57H、23H、-5H、-13H，正数的和为 00FBH，负数的和为-199H，对应补码为 FE67H。实验的结果如图 7 所示。结果图中，片内 RAM 地址 2CH 储存了正数低八位 FBH，2DH 存储了正数高位和符号位 00H；2EH 存储了负数低八位 67H，2FH 存储了负数高位和符号位 FEH，故运行结果正确。

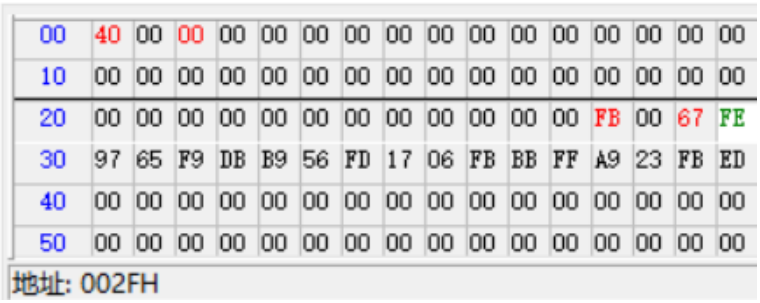


图 7 实验三设计实验 3 第一次实验结果

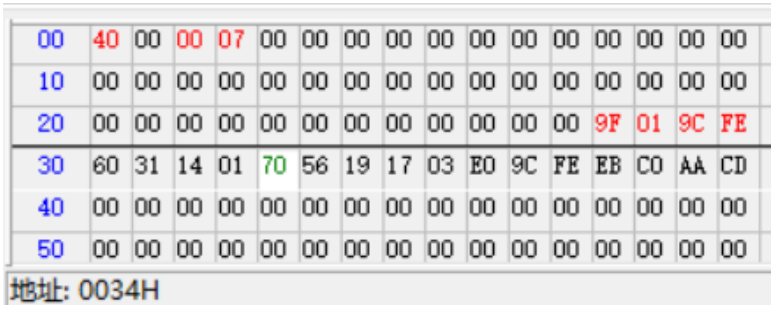


图 8 实验三设计实验 3 第二次实验结果

第二次实验，写入数据：60H、31H、14H、01H、70H、56H、19H、17H、03H、-20H、-64H、-2H、-15H、-40H、-56H、-33H。实验结果如图 8 所示。这组数据正数的和为 019FH，负数的和为 FE9CH，它们被分别存储到了片内 RAM 地址 2CH~2FH 的单元内。故运行结果正确。

```
        ORG 0000H
        MOV R0, #30H
        MOV R2, #10H
        MOV R3, #00H
        MOV 2CH, #00H
        MOV 2DH, #00H
        MOV 2EH, #00H
        MOV 2FH, #0FFH
LOOP:   CLR C
        MOV A, @R0              ;取出数据
        JB ACC.7, NEG           ;判断符号，分别执行对应处理程序
POS:    ADD A, 2CH              ;正数低八位相加
        MOV 2CH, A              ;保存正数低八位
        MOV A, 2DH              ;取出正数高八位
        ADDC A, #00H            ;有进位则需改变高八位
        MOV 2DH, A
        LJMP LAST               ;处理完毕
NEG:    ADD A, 2EH              ;负数低八位相加
        INC R3
        MOV 2EH, A              ;储存负数低八位
        MOV B.0, C              ;保存进位标志
        CJNE R3, #01H, STEP     ;第一次负数加法运算需进行调整
        SETB C                  ;第一次负数加法运算缺少一个借位 C
        MOV A, 2FH
        ADDC A, #0FFH           ;补上借位 C，改变高八位
        MOV 2FH, A
        LJMP LAST
STEP:   MOV A, 2FH              ;其他次数负数加法，取出高八位
        MOV C, B.0              ;恢复借位
        ADDC   A, #0FFH         ;有借位则需改变高八位
        MOV 2FH, A              ;保存高八位
        LJMP LAST               ;处理完毕
LAST:   INC R0                  ;改变源地址
        DJNZ R2, LOOP           ;循环
        SJMP $
        END
```

Code 3 实验三设计实验 3 程序

实验四

1、实验目的

掌握比较指令的使用及循环程序的编写方法；掌握字符查找的思路和算法；理解并能运用查表和散转指令。

2、基础实验部分

（1）完成共阴数码管数值显示译码的功能，在 WAVE 环境运行程序，观察寄存器及内存单元的变化。

累加器 A 的值依次变为：3FH、06H、5BH、4FH、66H、6DH、7DH、07H、7FH、6FH、77H、7CH、58H、5EH、79H、71H。DPTR 从 000DH 递增到 001DH。

运行结束后程序 ROM 中的数据如图 9 所示。从地址 0DH 开始到 1EH，ROM 中存储的内容为表 TBL 的内容。

0000	7A	10	90	00	0D	74	00	93	A3	DA	FA	80	FE	3F	06	5B
0010	4F	66	6D	7D	07	7F	6F	77	7C	58	5E	79	71	00	40	FF
0020	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF
0030	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF
0040	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF
0050	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF

地址: 000DH

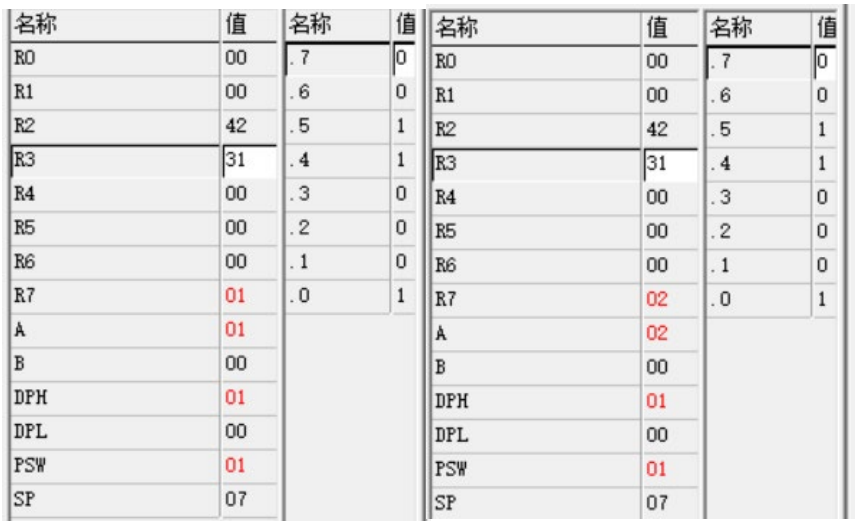
图 9 实验四基础实验 1 程序 ROM 数据

（2）完成一个两位十六进制数到 ASCII 码的转换，数值存放在 R2 中，转换结果低位存于 R2,高位存于 R3。用 PC 做基址实现。实验结果如图 10 所示。寄存器 R2 保存了低八位 B 的 ASCII 码‘B’（42H），R3 保存了高八位 1 的 ASCII 码‘1’（31H）。

名称	值	名称	值
R0	00	.7	0
R1	00	.6	0
R2	42	.5	1
R3	31	.4	1
R4	00	.3	0
R5	00	.2	0
R6	00	.1	0
R7	31	.0	1
A	31		
B	00		
DPH	00		
DPL	00		
PSW	01		
SP	01		

图 10 实验四基础实验 2 寄存器数据

（3）完成 256 字节范围内程序散转的功能，根据 R7 的内容转向各个子程序，在 WAVE 环境运行程序，观察寄存器及内存单元的变化。变化的情况如图 11 所示。累加器 A 的值变为 R7 设置的值，如图中当 R7 设置为 01H 时，执行了程序 PROG1，累加器 A 变为 01H；R7 设置为 02H 时，执行了程序 PROG2，累加器 A 变为 02H。



名称	值	名称	值
R0	00	.7	0
R1	00	.6	0
R2	42	.5	1
R3	31	.4	1
R4	00	.3	0
R5	00	.2	0
R6	00	.1	0
R7	01	.0	1
A	01		
B	00		
DPH	01		
DPL	00		
PSW	01		
SP	07		

名称	值	名称	值
R0	00	.7	0
R1	00	.6	0
R2	42	.5	1
R3	31	.4	1
R4	00	.3	0
R5	00	.2	0
R6	00	.1	0
R7	02	.0	1
A	02		
B	00		
DPH	01		
DPL	00		
PSW	01		
SP	07		

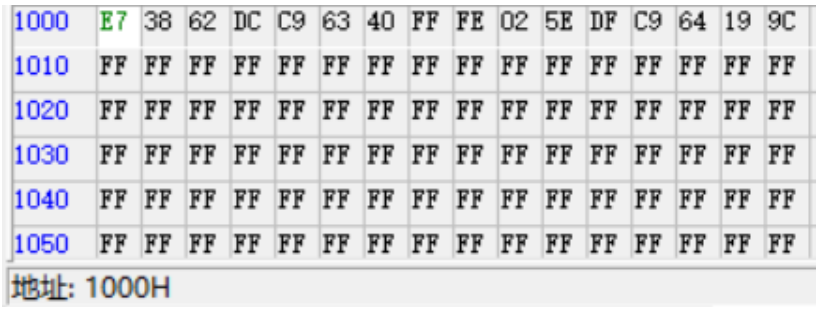
图 11 实验四基础实验 3 寄存器变化情况

3、设计实验部分

（1）在外部 RAM 1000H 开始处有 10H 个带符号数，请找出其中的最大值和最小值，分别存入内部 RAM 的 MAX、MIN 单元。程序如 Code4 所示，实验结果如图 12~图 13 所示。

00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
10	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
20	64	9C	00	00	00	00	00	00	00	00	00	00	00	00	00	00
30	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
40	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
50	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00

图 12 实验四设计实验 2 片内 RAM 数据



1000	E7	38	62	DC	C9	63	40	FF	FE	02	5E	DF	C9	64	19	9C
1010	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF
1020	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF
1030	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF
1040	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF
1050	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF

地址: 1000H

图 13 实验四设计实验 2 片外 RAM 数据

```
        MAX EQU 20H
        MIN EQU 21H
        ORG 0000H
        MOV DPTR, #1000H
        MOV R2, #10H
        MOVX A, @DPTR
        MOV MAX, A
        MOV MIN, A
LOOP:   MOVX A, @DPTR
        MOV B, MAX
        MOV C, ACC.7
        ANL C, /B.7             ;新的数为负数而 MAX 为正数
        JC MAXPS
        MOV C, B.7
        ANL C, /ACC.7           ;新的数为正数而 MAX 为负数
        JC CGE1
        CLR C
        SUBB A, MAX
        JC MAXPS                ;(A)<(MAX)
        MOVX A, @DPTR
CGE1:   MOV MAX, A
        LJMP MAXPS
MAXPS:  MOVX A, @DPTR
        MOV B, MIN
        MOV C, ACC.7
        ANL C, /B.7             ;新的数为负数而 MIN 为正数
        JC CGE2
        MOV C, B.7
        ANL C, /ACC.7           ;新的数为正数而 MIN 为负数
        JC MINPS
        CLR C
        MOVX A, @DPTR
        SUBB A, MIN
        JNC MINPS               ;(A)>(MIN)
        MOVX A, @DPTR
CGE2:   MOV MIN, A
        LJMP MINPS
MINPS:  INC DPTR
        DJNZ R2, LOOP
        SJMP $
        END
```

Code 4 实验四设计实验 2 程序

可以观察到，程序运行结束后，片内 RAM 地址 20H 存储了最大值 64H，21H 存储了最小值 9C，故运行结果正确。

（2）分别用近程查表指令和远程查表指令，查找 R3 内容的平方值。平方值为两个字节数据。近程查表程序如 Code5 所示，实验结果如图 14 所示。

```
        ORG 0000H
        MOV R3, #27
        MOV A, R3
        DEC A
        RL A                ;调整 A 以适应表中数据分布（一个平方数 2 字节）
        MOV R2, A           ;保存 A
        ADD A, #08H         ;修正 A
        MOVC A, @A+PC       ;查表得高八位
        MOV R6, A           ;储存高八位     1BYTE
        MOV A, R2           ;恢复 A         1BYTE
        ADD A, #04H         ;修正 A         2BYTE
        MOVC A, @A+PC       ;查表得低八位   1BYTE
        MOV R7, A           ;储存低八位     1BYTE
        SJMP $              ;               2BYTE
TABLE:  DW 1,4,9,16,25,36,49,64,81,100
        DW 121,144,169,196,225,256,289,324,361,400
        DW 441,484,529,576,625,676,729,784,841
        END
```

Code 5 实验四设计实验 3 近程查表程序

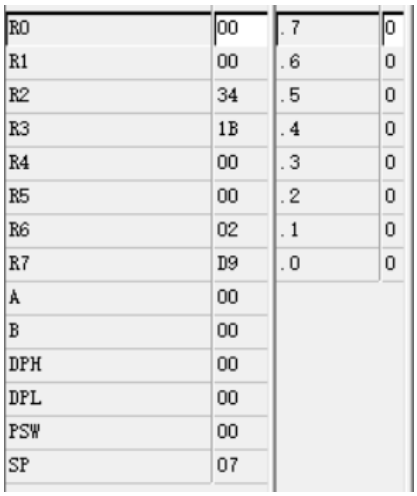


图 14 实验四设计实验 3 近程查表寄存器数据

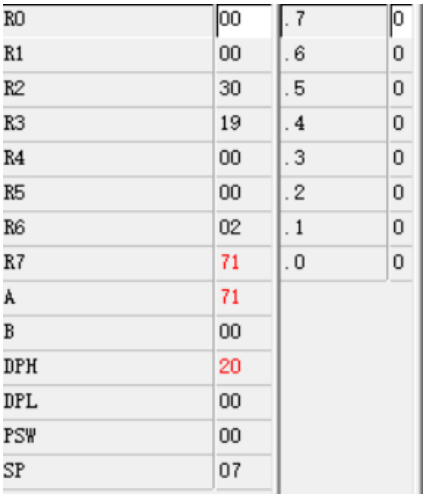
当使用近程查表时，需调整累加器 A 的值，这里增加的值为近程查表指令和表头之间的指令的字节数。程序中各指令的字节数已在注释中列出，故第一次查表累加器 A 需加上 08H，第二次查表需加上 04H。程序运行结束后，寄存器 R6 保存了平方的高八位，R7 保存了低八位，平方的结果为 02D9H=729D=$27D^2$，故程

序运行结果正确。

远程查表程序如 Code6 所示，实验结果如图 15 所示。

```
        ORG 0000H
        MOV DPTR, #TABLE
        MOV R3, #25
        MOV A, R3
        DEC A
        RL A                ;调整 A 以适应表中数据分布（一个平方数 2 字节）
        MOV R2, A           ;保存 A
        MOVC A, @A+DPTR     ;查表得高八位
        MOV R6, A           ;储存高八位
        MOV A, R2           ;恢复 A
        INC A               ;修正 A
        MOVC A, @A+DPTR     ;查表得低八位
        MOV R7, A           ;储存低八位
        SJMP $
        ORG 2000H
TABLE:  DW 1,4,9,16,25,36,49,64,81,100
        DW 121,144,169,196,225,256,289,324,361,400
        DW 441,484,529,576,625,676,729,784,841
```

Code 6 实验四设计实验 3 远程查表程序



R0	00	.7	0
R1	00	.6	0
R2	30	.5	0
R3	19	.4	0
R4	00	.3	0
R5	00	.2	0
R6	02	.1	0
R7	71	.0	0
A	71		
B	00		
DPH	20		
DPL	00		
PSW	00		
SP	07		

图 15 实验四设计实验 3 远程查表

寄存器数据

程序运行结束后，寄存器 R6 保存了平方的高八位，R7 保存了低八位，平方的结果为 0271H=625D=25D^2，故程序运行结果正确。

实验总结与心得

本次实验主要熟悉了 51 微控制器指令，掌握了算术运算、逻辑运算、字符和进制转换、查表等相关操作技巧。重点理解了有符号数的运算，涉及到了对状态寄存器 PSW 的 Cy 和 OV 位的判断；以及位运算、位判断，涉及到位寻址相关

指令，这些在实验前掌握得相对生疏。

本次实验遇到的主要问题是在进行实验三设计实验 3 时，负数的累加求和一直无法得到正确的结果。后来通过单步调试的方法，注意到了对第一个负数进行累加时，进行的运算是负数加 0，在进行补码加法时会导致缺少一个进位，使得 Cy=0，最终结果的高八位（含符号位）不正确。因此在第一次负数加法时额外加 1，就可以使结果正确。