浙江大学实验报告

专业:信息工程 姓名:李坤林

学号: 3200101135 日期: 11/10/2022

地点:外经贸楼实验室

课程名称:微机原理与接口技术 指导老师:黄凯 实验名称:内存操作和数制及代码转换

一、实验目的和要求

算术运算:

- ① 掌握算术运算类、逻辑运算类指令的使用方法;
- ② 掌握 BCD 码、补码数制表示方法;
- ③ 掌握运算程序及循环程序的编写和调试方法。

比较和杳表:

- (1) 掌握比较指令的使用及循环程序的编写方法;
- ② 掌握字符查找的思路和算法;
- ③ 理解并能运用查表和散转指令。

二、实验内容和原理

比较和杳表:

- ① 以下程序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,将变化结果注释于右侧。
- ② 以下子程序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 结果地位存于 R2,高位存于 R3。用 PC 做基址实现。
- ③ 以下程序完成 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,将变化结果注释于右侧。
- ④ 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。R3 内容小于等于 0FH,即平方值为单字节数据。
- ⑤ 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。
- ⑥ 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。数制及代码转换:
- ① 以下程序完成单字节的 BCD 码加法功能,完成空白处程序填写,并在 WAVE 环境 运行程序,观察寄存器及内存单元的变化。
- ② 下列程序完成多字节 BCD 码加法运算。内部 RAM30H 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)。
- ③ 设计程序,实现任意字节压缩 BCD 码的相加,使用单步、断点方式调试程序,查 看结果。
- ④ 设计程序, 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, 查看结果。
- ⑤ 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)。请分别在 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。

⑥ 设计程序,实现十六进制数双字节乘单字节 35A6H*56H,结果存于 40H 开始的三 个单元中,使用单步、断点方式调试程序,查看结果。

三、主要仪器设备

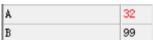
计算机一台

四、操作方法和实验步骤

Lab3 算术运算

Code1

A 99, B 99, 十六进制相加得 132, 加完 A 为 32 (10011001+10011001)



低四位高四位全大于 9, DA 语句均要+6 处理, 所以 A=98H A 98

ADDC 后, CY+00+A 自己得 0

LJMP 高位处理后: 30 98 01 0

Code2

RO

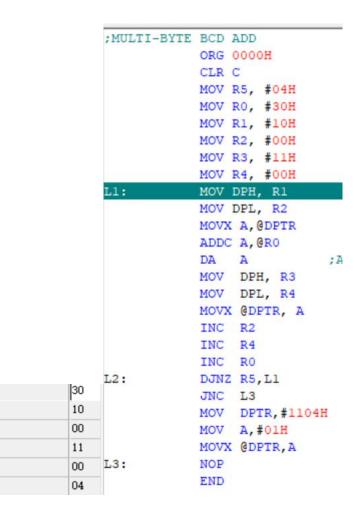
R1

R2

R3

R4

R5



R5 是循环次数,每次到 DJNZ 就减 1,直到为 0;

RO 存放加数,每次执行 INC,自增 1

R1 和 R2 是被加数得高位和低位,每次执行到 INC, R2 自增,R1 不变R3 和 R4 是和得高位和低位,执行到 INC,R4 自增,R3 不变;规定得1100-1103 是存放和的,进位存在 04 里。

```
1100 65 66 66 66 01
```

CODE3

实现 543210H 和 9876 的相加, 543210H+9876H=553086H

```
Compressed BCD CODE
ORG 0000H
MOV 3DH, #54H
MOV 3FH, #32H
MOV 3FH, #32H
MOV 4FH, #76H
AJMP MAIN
ORG 0100H
MAIN: MOV R3, #03H ;BCD BYTE LENGTH
MOV R0, #3FH
MOV DFTR, #1000H
CLR C
BCD: MOV A, @RO
ADDC A, @RI
ADA A ;TEN ADJUSTMENT
MOV R1, #4FH
MOV DFTR, # ;STORE THE SUM_BCD
DEC R1
INC DFTR
DJNZ R3, BCD
JNC NEXT
MOV R, #01H
MOVX R0 PFTR, A ;CY
NEXT: MOV R3, #03H
MOV R0, #51H
MOV R0, #51H
MOV R0, #51H
MOV R0, #37H
MOV BR0, A
INC DFTR
DDC R0
DJNZ R3, LOOP
SJMF S
END
```

Code4

123456H-005634H

```
:123456H-005634H
               RESULT EQU
                               32H
               ORG
                        0000H
                AJMP
                        MAIN
                        0100H
MAIN:
                CLR
               MOV
                        RO, #12H
               MOV
                        R1, #34H
               MOV
                        R2, #56H
               MOV
                        R3, #00H
               MOV
                        R4, #56H
               MOV
                        R5, #34H
               MOV
                        A, R2
                SUBB
                MOV
                        RESULT, A
                MOV
                SUBB
                        A, R4
                MOV
                        RESULT-1, A
               MOV
                        A, RO
                SUBB
                       A. R3
                        RESULT-2, A
               MOV
                SJMP
```

储存在 30-32H, 结果是 11DE22H

30 11 DE 22

根据前面 RESULT 分配的位置,可以改变不同的结果储存位置

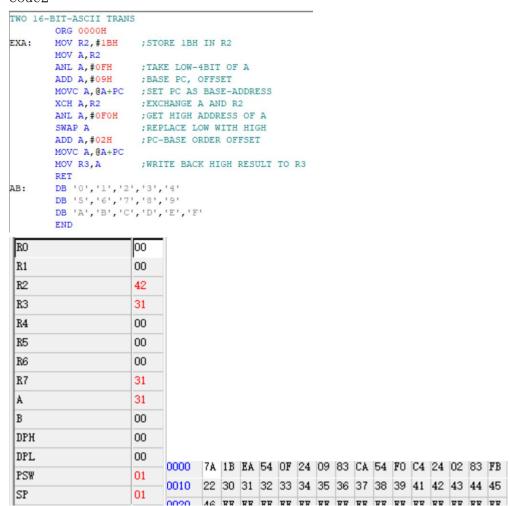
Lab4 查表

Code1



ROM 从 0013H 开始存入表格,内部寄存器,R2 循环 10H,最终值为 0; A 最终值为 71H,是 TBL 中第 17 个数。

Code2



转换后的结果存在 R2、3 中, R2 存低位 B, R3 存高位 A, 分别为 42, 31; ROM 中从 0012H 开始存入 0-F 的 ASCII 码 31-45;

Code3

```
;256 BIT PROGRAM Program Dispersion
         ORG 0000H
         MOV DPTR, #TAB ; GIVE TAB HEAD ADDRESS TO DPT
START:
         MOV A, R7
                        ;R7*2 TO MATCH MACHINE CODE O
         JMP @A+DPTR
                       ; IF AJMP IN TAB TRANS TO LJMP
         ORG 0100H
TAB:
         AJMP PROGO
                       ; JUMP TO SELECTED ADDRESS
         AJMP PROG1
         AJMP PROG2
         AJMP PROG3
         SJMP $
PROG0:
         MOV A, #00H
                        ;GIVE VALUE A
         SJMP RE
PROG1:
         MOV A.#01H
         SJMP RE
PROG2:
         MOV A, #02H
         SJMP RE
PROG3:
         MOV A, #03H
RE:
         END
```

先将 R7 中的值赋给 A, 再将 A 中数据变大二倍当作指令的偏移量; DPTR 是散转表的起始地址;通过 JUMP@A+DPTR,寻找对应指令,然后跳转到相应 指令,对 A 进行赋值。最终达成将 R7 的值写入 A。

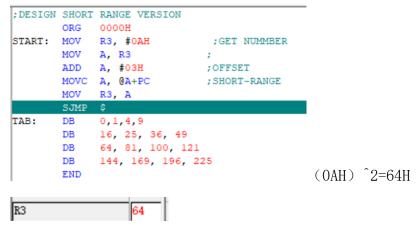


Code4

都是查表指令,MOVC A, @A+PC, 只能给累加器 A 赋值, 所以只能查这条指令所在地址以后 256 字节范围内的代码或常数。

而 MOVC A, @A+DPTR,可以给 DPTR 赋给任何一个 16 位的地址值,所以查表范围可达整个程序存储器 64K 字节空间的代码或常数。

1. 近程



2. 远程

```
; LONG RANGE
         ORG 0000H
STASRT:
         MOV DPTR, #TAB
         MOV R3, #OAH
         MOV A, R3
         MOVC A. @A+DPTR
         MOV R3, A
         SJMP $
              0, 1, 4, 9
TAB:
         DB
         DB
             16, 25, 36, 49
         DB 64, 81, 100, 121
         DB 144, 169, 196, 225
         END
   64
R3
```

Code5

```
MAX
       EQU
               30H
MIN
      EQU
              31H
       ORG 0000H
                        ;LOOP TIME
       MOV R7, #OFH
       MOV DPTR, #1000H
                          ;DPTR->START ADDRESS OF XRAM
       MOVX A, @DPTR
                        ;STORE DATA IN A
       ADD A, #80H
                         ; MAKE IT NON-SIGNED NUMBER
       MOV MAX, A
                         ; INITIALIZED MAX
       MOV MIN, A
                          ; INITIALIZED MIN
L1:
       INC DPTR
                          ; READ IN NEXT NUMBER
       MOVX A, @DPTR
       ADD A, #80H
       CLR C
                        ;SUB IT, IF !=, TO S1, IF A<MAX, C->1
       CJNE A, MAX, S1
                          ; IF =, JUMP TO NEXT
       SJMP NEXT
                         ; IF C=1, MEANS A<MAX, TO S2
S1:
       JC S2
       MOV MAX, A
                          ; UPDATE MAX
       SJMP NEXT
                        ;!=, TO S2, IF A<MIN SAME TIME, C->1
       CJNE A, MIN, S3
53:
      JNC NEXT
                         ; IF C=0, MEANS A>MIN, TO NEXT
       MOV MIN, A
                         ; ELSE, UPDATE MIN
NEXT: DJNZ R7, L1
                          ; JUDGE LOOP TIME, IF NOT TO OFH, CONTINUE
       MOV A, MAX
       SUBB A, #80H
       MOV MAX, A
                         GET MAX
       MOV A.MIN
       SUBB A, #80H
       MOV MIN, A
                        GET MIN
       SJMP $
       END
```