《微机实验》报告

实验名称 Keil 的使用与汇编语言上机操作 成 绩

指导教师 曹丹华

专业班级 光电 1802 姓 名 马笑天 学 号 U201813654

联系电话 18201000620

一、任务要求

1.掌握 Keil 环境的使用

1）字节拆分、合并：调试 e421.asm 程序， 观察相关寄存器和单元的内容。

2）数据块填充：调试 fill.asm 程序，观察相关寄存器和单元的内容。

2. 编写两个十六位数的加法程序。

有两个十六位无符号数，分别存放在从 20H 和 30H 开始的数据区中，低八位先存，高八位在后，和存于 R3（高八位）和 R4（低八位），进位位存于 R2。

二、设计思路

题目要求设计两个十六位无符号数的相加，原始数据的存储空间 20H、21H，以及 30H、31H，低八位先存，高八位在后。所求结果中，高八位存于 R3，低八位存于 R4，进位位存于 R2。

在低八位的计算时，考虑使用加法指令 ADD，注意开始计算前将 C 清零；之后的高八位，考虑到有可能存在来自低八位的进位，我们使用带进位的加法指令 ADDC；最后考虑到可能存在的来自高八位的进位，进位位也应另行考虑！本题中存在了 R2 中。这样就能保证各种测试数据都能够正常运行了！

三、资源分配

ROM：

代码从 0000H 开始

片上 RAM：

20H、21H：第一个十六位无符号数，其中 20H 存入低八位，21H 存入高八位

30H、31H：第二个十六位无符号数，其中 30H 存入低八位，31H 存入高八位

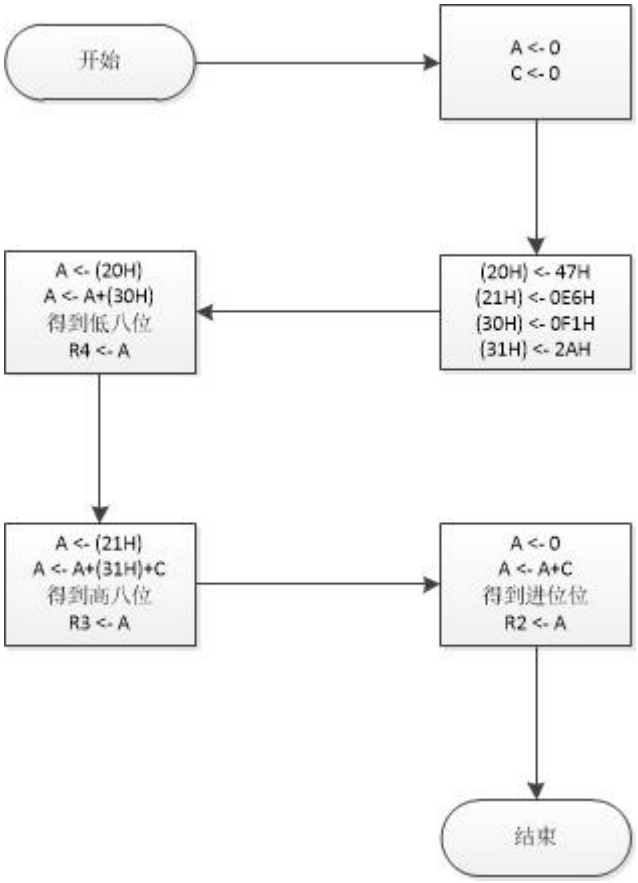
累加器 A：当作中间临时存放空间

R3、R4：分别是最后计算结果的高八位和低八位存放空间

C：进位标志位，用于指示是否存在来自高八位的进位

R2：用于指示是否存在来自高八位的进位

四、流程图



五、源代码 （含文件头说明、语句行注释）

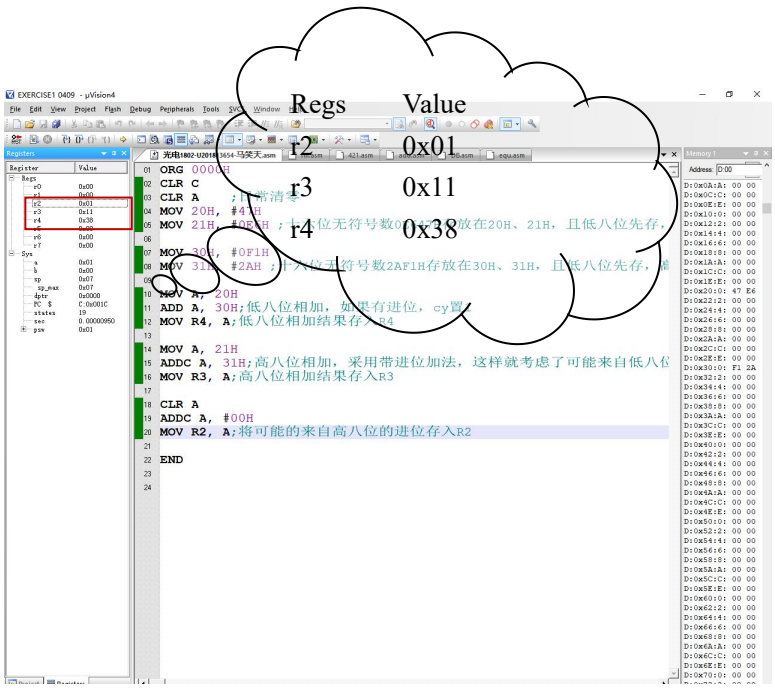
```
;Filename:      add.asm
;Function:      利用 ADD/ADDC 指令，实现双字节加法，带有高位进位保存功能
;Date:          2020.4
;Designed by:   MA Xiaotian
;Source used:   20H,21H,30H,31H: two 16-bit unsigned int
                R3, R4: for the result, R3 for the upper 8 bit, R4 for
                the lower 8 bit
                R2: for the carry bit
        ORG 0000H
        LMPJ MAIN
        ORG 2000H
MAIN:   CLR C
        CLR A    ;日常清零
        MOV 20H, #47H
        MOV 21H, #0E6H ;十六位无符号数 0E647H 存放在 20H、21H，且低八位先，高
    八位在后

        MOV 30H, #0F1H
        MOV 31H, #2AH ;十六位无符号数 2AF1H 存放在 30H、31H，且低八位先存，高
    八位在后

        MOV A, 20H
        ADD A, 30H;低八位相加，如果有进位，Cy 置 1
        MOV R4, A;低八位相加结果存入 R4

        MOV A, 21H
        ADDC A, 31H;高八位相加，采用带进位加法，这样就考虑了可能来自低八位的
```

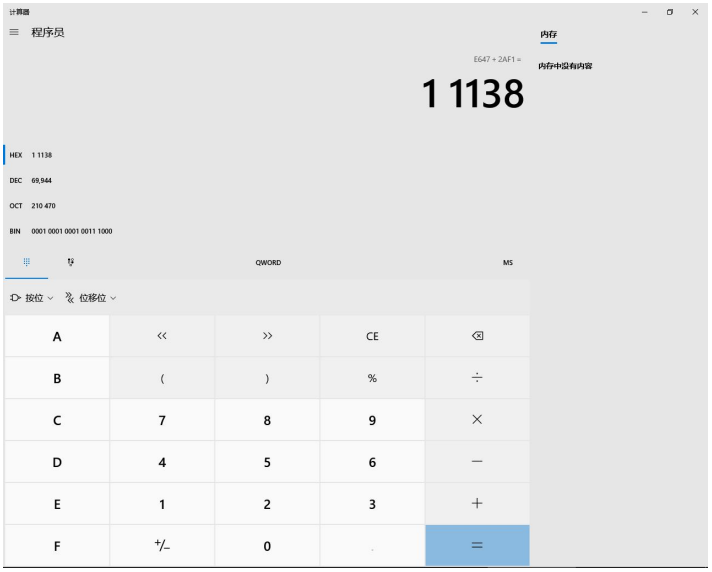
进位；同样的，如果相加结果有进位，则 Cy 置 1

```
MOV R3, A;高八位相加结果存入 R3


//关于 R2 的存入，下面给出两种方法
//方法一：（是在上机课时和助教以及同学讨论的方案）
CLR A
ADDC A, #00H
MOV R2, A;将可能的来自高八位的进位存入 R2
//方法二：在执行时请手动去除下面两行的注释标记
//MOV 10H, C;通过位传输指令、位寻址的方式，将 Cy 存入 22H 单元中
//MOV R2, 22H

END
```

六、程序测试方法与结果

程序测试的方法：采用了一组结果含有进位位的测试数据进行检验。结果如下图所示，可以看到，这段代码能够正常的编译运行，并且能很好地完成既定任务。

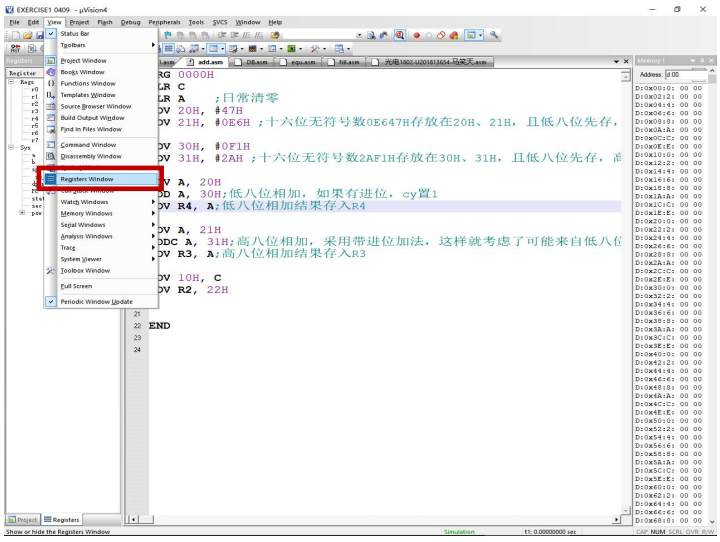




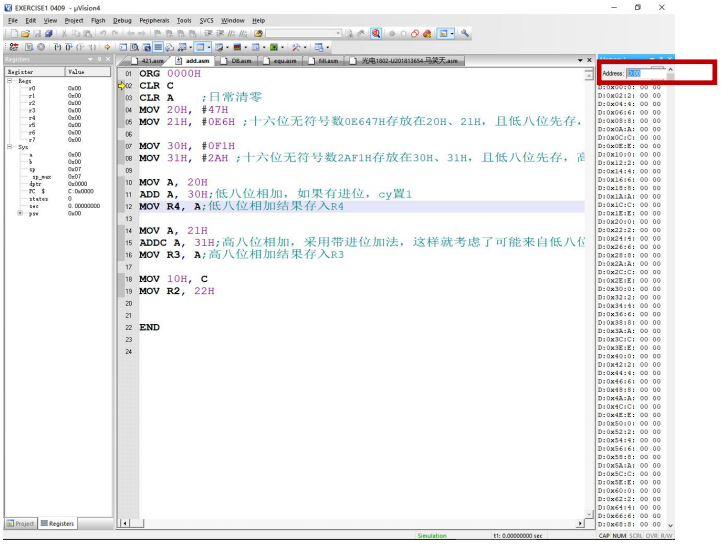
思考题

1. 怎样查看工作寄存器、SFR、片内 RAM、片外 RAM 及程序代码空间内容？Disassembly 窗口有何作用？

答：查看工作寄存器的内容，需要在工具栏中 Translate->Build->Start/Stop Debug Session 的前提下，再在标题栏中按照 View->Registers Window 的窗口调出；



查看 SFR、片内 RAM、片外 RAM 及程序代码空间的内容，需要在工具栏中 Translate->Build->Start/Stop Debug Session 的前提下，再在标题栏中按照 View->Memory Windows->Memory 1 的顺序调出内存窗口。因为 SFR、片内 RAM 同属片内 RAM 区，若查看则在 Address 窗口键入“D:00”即可查看。同样的，由于片外 RAM 与程序代码空间分别存储在片外 RAM 区和 ROM 区，若查看则应分别在 Address 窗口键入“X:00”、“C:00”即可查看。



Disassembly 窗口的作用是反汇编，便于程序员了解代码的位置以及在内存中的表示、指令字节长度等重要信息。

2. 字节拆分、合并还有哪些方法，举一例说明。

答：可以利用 ADD 指令。

例如：

```
        ORG 0000H
        LJMP MAIN
        ORG 1000H
MAIN:   MOV A, #98H
        MOV 20H, A
        ANL A, #0FH;    取 98H 的低四位
        MOV A, 21H;     低四位存入 21H
        MOV A, 20H
```

```
        ANL A, #0F0H;    取 98H 的高四位
        MOV A, 22H;      高四位存入 22H
        MOV 22H, A
        ADD A, 21H;      合并
        MOV A, 23H       合并结果存入 23H
```

3. 若按递减 1 规律填充数据块，应如何修改程序？

答：

```
        ORG 0000H
        LJMP MAIN

        ORG  0100H
MAIN:   MOV SP,#40H

FILL:   //CLR  A                    ;A 寄存器清零
        //MOV  R0,  #00H             ;设循环计数器
        MOV R0, #FFH; 设循环计数器
        MOV  DPTR,  #7000H          ;设数据指针
FILL1:  //MOVX  @DPTR,  A           ;传送到片外 RAM
        MOVX  @DPTR,  R0            ;传送到片外 RAM
        //INC  A                    ;A 内容加 1
        INC  DPTR                   ;修改数据指针
        //INC  R0                   ;修改循环计数器
        //CJNE  R0,  #00H,  FILL1   ;判断是否结束
        DJNZ R0, FILL1              ;判断是否结束
HERE:   SJMP HERE                   ;原地踏步
        END
```

4. 若从 7010H 单元开始，连续填充 20 个字节，应该如何修改程序？

答：

```
            ORG 0000H
            LJMP MAIN

            ORG  0100H
MAIN:       MOV SP,#40H

FILL:       CLR  A                      ;A 寄存器清零
            MOV  R0,  #00H              ;设循环计数器
            //MOV  DPTR,  #7000H        ;设数据指针
            MOV DPTR, #7010H
FILL1:      MOVX  @DPTR,  A             ;传送到片外 RAM
            INC  A                      ;A 内容加 1
            INC  DPTR                   ;修改数据指针
            INC  R0                     ;修改循环计数器
            //CJNE  R0,  #00H,  FILL1       ;判断是否结束
            CJNE R0, #10H, FILL1        ;判断是否结束
HERE:       SJMP HERE                   ;原地踏步
            END
```

5. 若完成双字节 BCD 码加法，应如何修改程序？

答：

```
        ORG 0000H
        LJMP MAIN:
        ORG 2000H
MAIN:    CLR C
```

```
CLR A     ;日常清零
MOV 20H, #38H
MOV 21H, #49H ;十六位无符号数 0E647H 存放在 20H、21H，且低八位先存，高八位在后

MOV 30H, #25H
MOV 31H, #17H ;十六位无符号数 2AF1H 存放在 30H、31H，且低八位先存，高八位在后

MOV A, 20H
ADD A, 30H;低八位相加，如果有进位，Cy 置 1
DA A          ;二-十进制调整指令
MOV R4, A;低八位相加结果存入 R4

MOV A, 21H
ADDC A, 31H;高八位相加，采用带进位加法，这样就考虑了可能来自低八位的进位；同样的，如果相加结果有进位，则 Cy 置 1
DA A          ;二-十进制调整指令
MOV R3, A;高八位相加结果存入 R3
//关于 R2 的存入，下面给出两种方法
//方法一：（是在上机课时和助教以及同学讨论的方案）
CLR A
ADDC A, #00H
MOV R2, A;将可能的来自高八位的进位存入 R2
//方法二：在执行时请手动去除下面两行的注释标记
//MOV 10H, C;通过位传输指令、位寻址的方式，将 Cy 存入 22H 单元中
//MOV R2, 22H

END
```

本人承诺:

本报告内容真实，无伪造数据，无抄袭他人成果。本人完全了解学校相关规定，如若违反，愿意承担其后果。

签字：马笑天

2020 年 4 月 14 日