山东大学

信息科学与工程学院

2020－2021 学年第二学期

实 验 报 告

课程名称： 微处理器原理与应用

实验名称： 实验 3 完成三个程序的编写

专 业 班 级 2019 级崇新学堂

学 生 学 号 201900121023

学 生 姓 名 李禹申

实 验 时 间 2021 年 4 月 4 日

实验报告

【实验目的】

通过对三个程序的编写，在苦痛中浴火重生完成涅槃

【实验要求】

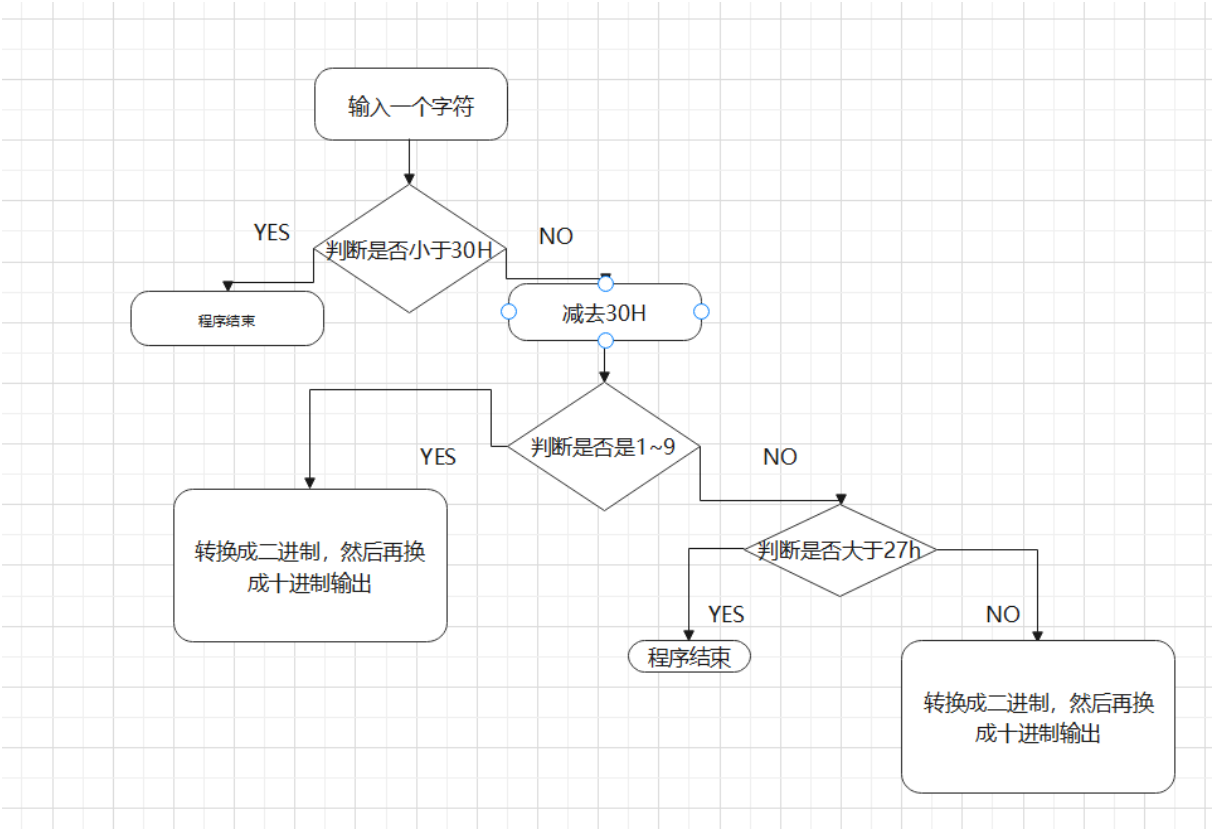
认真编写实验，并完成注释

【实验具体内容】

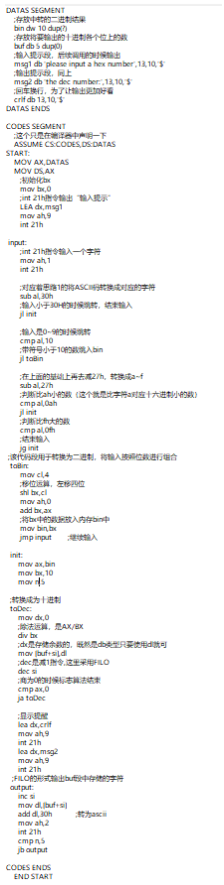
1. 编写十六进制转换成十进制的程序
2. 理解并注释判断闰年程序
3. 实现 3+5
4. 进行多位数相加减

【第一个实验】实现十进制转换成为十六进制

（1）实验流程图（从实验 2.2 分支程序实验和循环程序实验开始必须画流程图）：



（2）实验源代码（粘贴源代码）：



```
DATAS SEGMENT
    ;存放中转的二进制结果
    bin dw 15 dup(?)
    ;存放将要输出的十进制各个位上的数
    buf db 10 dup(0)
    ;输入提示段，后续调用的时候输出
    msg1 db 'please input a hex number',13,10,'$'
    ;输出提示段，同上
    msg2 db 'the dec number:',13,10,'$'
    ;回车换行，为了让输出更加好看
    crlf db 13,10,'$'
DATAS ENDS

CODES SEGMENT
    ;这个只是在编译器中声明一下
    ASSUME CS:CODES,DS:DATAS
START:
    MOV AX,DATAS
    MOV DS,AX
      ;初始化bx
      mov bx,0
      ;int 21h指令输出 "输入提示"
      LEA dx,msg1
      mov ah,9
      int 21h

input:
      ;int 21h指令输入一个字符
      mov ah,1
      int 21h

      ;对应着思维1的将ASCII码转换成对应的字符
      sub al,30h
      ;输入小于30h的时候跳转，结束输入
      jl int

      ;输入是0-9的时候跳转
      cmp al,10
      ;带符号小于10的数就输入bin
      jl toBin

      ;在上面的基础上再去减27h，转换成a-f
      sub al,27h
      ;判断比ah小的数（这个就是处理ah对应十六进制小的数）
      cmp al,0ah
      jl int
      ;判断比大大的数
      cmp al,0fh
      ;结束输入
      jg int
;操作的说明:用于转换为二进制，将输入按照位数进行组合
toBin:
      mov cl,4
      ;移位运算，左移四位
      shl bx,cl
      mov ah,0
      add bx,ax
      ;将bx中的数据放入内存bin中
      mov bin,bx
      jmp input       ;继续输入

int:
      mov ax,bin
      mov bx,10
      mov si,0

;转换成为十进制
toDec:
      mov dx,0
      ;除法运算，是AX/BX
      div bx
      ;dx是存储余数的，既然是dx余至只要使用dl就可
      mov [buf+si],dl
      ;dec是减1指令,这里采用引LO
      dec si
      ;商为0时候标志算法结束
      cmp ax,0
      ja toDec

      ;显示提醒
      lea dx,crlf
      mov ah,9
      int 21h
      lea dx,msg2
      mov ah,9
      int 21h
;引LOP的形式输出buf段中存储的字符
output:
      inc si
      mov dl,[buf+si]
      add dl,30h          ;转为ascii
      mov ah,2
      int 21h
      cmp si,5
      jb output

CODES ENDS
    END START
```

DATAS SEGMENT

;存放中转的二进制结果

bin dw 15 dup(?)

;存放将要输出的十进制各个位上的数

buf db 10 dup(0)

;输入提示段，后续调用的时候输出

msg1 db 'please input a hex number',13,10,'$'

;输出提示段，同上

msg2 db 'the dec number:',13,10,'$'

;回车换行，为了让输出更加好看

crlf db 13,10,'$'

DATAS ENDS

CODES SEGMENT

;这个只是在编译器中声明一下

ASSUME CS:CODES,DS:DATAS

START:

```
    MOV AX,DATAS
    MOV DS,AX
        ;初始化 bx
        mov bx,0
        ;int 21h 指令输出“输入提示”
        LEA dx,msg1
        mov ah,9
        int 21h

input:
        ;int 21h 指令输入一个字符
        mov ah,1
        int 21h

        ;对应着思路 1 的将 ASCII 码转换成对应的字符
        sub al,30h
        ;输入小于 30H 的时候跳转，结束输入
        jl init

        ;输入是 0~9 的时候跳转
        cmp al,10
        ;带符号小于 10 的数跳入 bin
        jl toBin

        ;在上面的基础上再去减 27h，转换成 a~f
        sub al,27h
        ;判断比 ah 小的数（这个就是比字符 a 对应十六进制小的数）
        cmp al,0ah
        jl init
        ;判断比 fh 大的数
        cmp al,0fh
        ;结束输入
        jg init
;该代码段用于转换为二进制，将输入按照位数进行组合
```

```
toBin:
        mov cl,4
        ;移位运算，左移四位
        shl bx,cl
        mov ah,0
        add bx,ax
        ;将 bx 中的数据放入内存 bin 中
        mov bin,bx
        jmp input              ;继续输入

init:
      mov ax,bin
      mov bx,10
      mov si,5

;转换成为十进制
toDec:
      mov dx,0
      ;除法运算，是 AX/BX
      div bx
      ;dx 是存储余数的，既然是 db 类型只要使用 dl 就可
      mov [buf+si],dl
      ;dec 是减 1 指令,这里采用 FILO
      dec si
      ;商为 0 的时候标志算法结束
      cmp ax,0
      ja toDec

      ;显示提醒
      lea dx,crlf
      mov ah,9
      int 21h
      lea dx,msg2
      mov ah,9
```

```
        int 21h
    ;FILO 的形式输出 buf 段中存储的字符
  output:
        inc si
        mov dl,[buf+si]
        add dl,30h                ;转为 ascii
        mov ah,2
        int 21h
        cmp si,5
        jb output
        ;这段和 CSDN 老鸟学的
        mov ah,1
        int 21h
        MOV AH,4CH
        INT 21H

CODES ENDS
    END START
CODES ENDS
    END START
```

（3）实验代码、过程、相应结果（截图）并对实验进行说明和分析：

【该程序设计到的思路（略有网络借鉴成分）】

汇编实现十进制转换成十六进制应该分以下几步：

First step：因为 DOS 读到的输入字符是用 ASCII 码表示的，也就是说首先要将 ASCII 转换成为对应的十六机制数字，具体实现方法是 1~9 的字符将 ASCII 码减去 30h，a~b 的字符减去 57h。注意！这个运算实际上贼坑！因为 16 进制的运算和平常 10 进制的运算习惯是不一样的。

Second step：移位运算，为什么要移位呢，因为在汇编指令使用 int 21H 的时候就只能一个字符、一个字符的输入，所以要将先输入进行移位运算，这属于处理多位数问题

Third step：接下来如何进行输出呢？首先要给程序一个我已经输入完的标志，一般都是 enter 键吧，这个时候就想到了上次实验 2.2 的判断字符问题，因此可以改写上次的程序完成对 enter 键的读取功能。

网络老鸟：1.*在改进自己的程序的时候发现了一个十分老鸟的写法，就是在判断输入为 enter 的时候使用 JL 这样使得 AX 中储存的东西变成了有符号数，然后跳转到 enter 块。*

但是后来再一看……发现自己想多了，就是 ASCII 码中对应字符小于 30H 的都会直接跳转

2.
```
add bx,ax
;将bx中的数据放入内存bin中
mov bin,bx
```
我在改进代码的时候，发现这个很是巧妙，就是现在 DATAS 数据段中开辟 memory 用来人为指定存储某些带有人类主观色彩的数据。

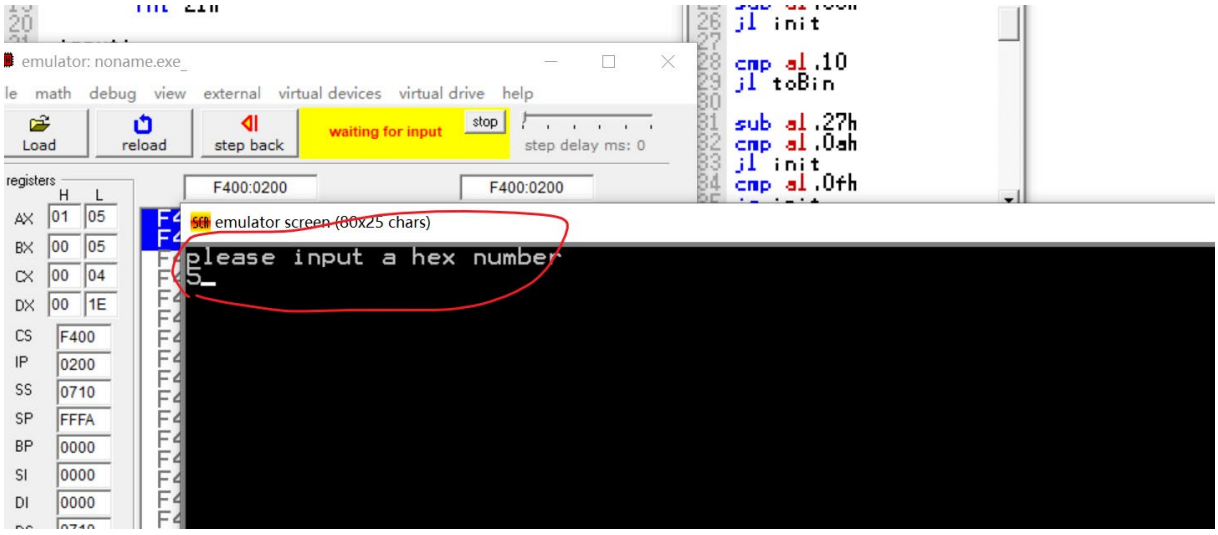
```
;存放中转的二进制结果
bin dw 10 dup(?)
;存放将要输出的十进制各个位上的数
buf db 5 dup(0)
```
其中由于 emu8086 软件的 bug 开辟 db 类型的空间是 dup 是不能使用"?"的。这个写法让我改进了我的程序也对 DATAS 有了理解，开始我是直接用[地址]的形式进行存储的

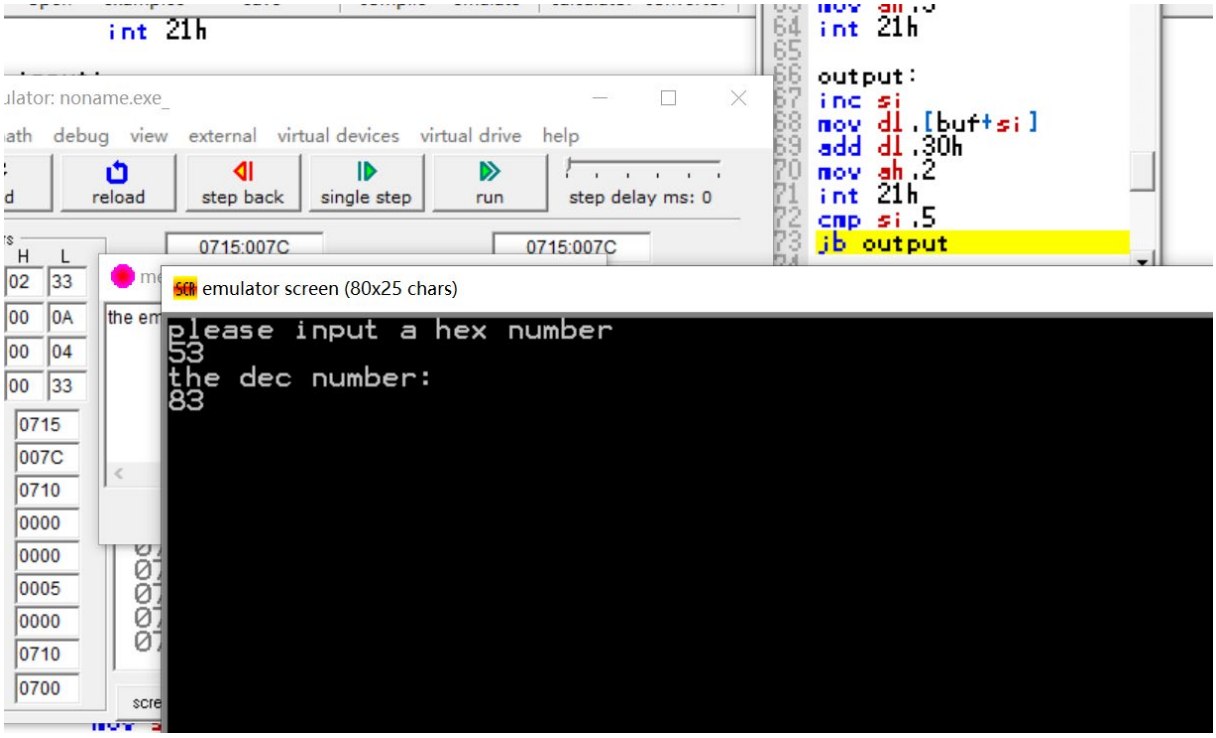
3.
```
mov ah,1
int 21h
MOV AH,4CH
INT 21H
```
这一个代码段是和 CSDN 的老鸟学的，使得每次运行程序不至于一下就停止

【程序执行展示】

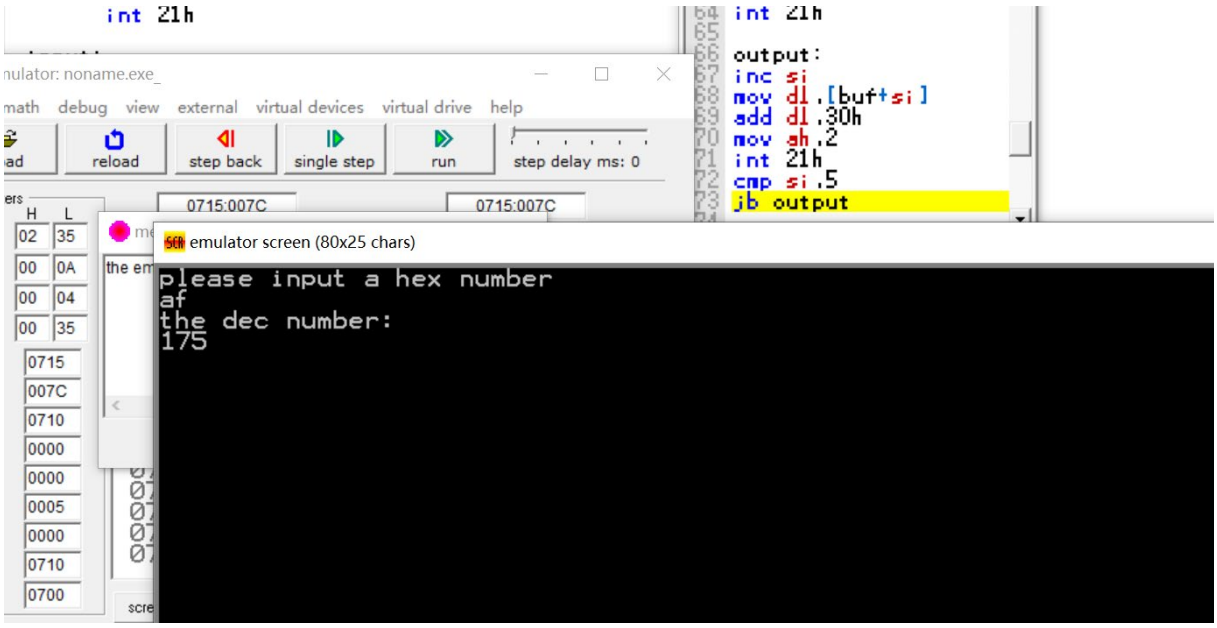
1.输入单个字符

①0~9：

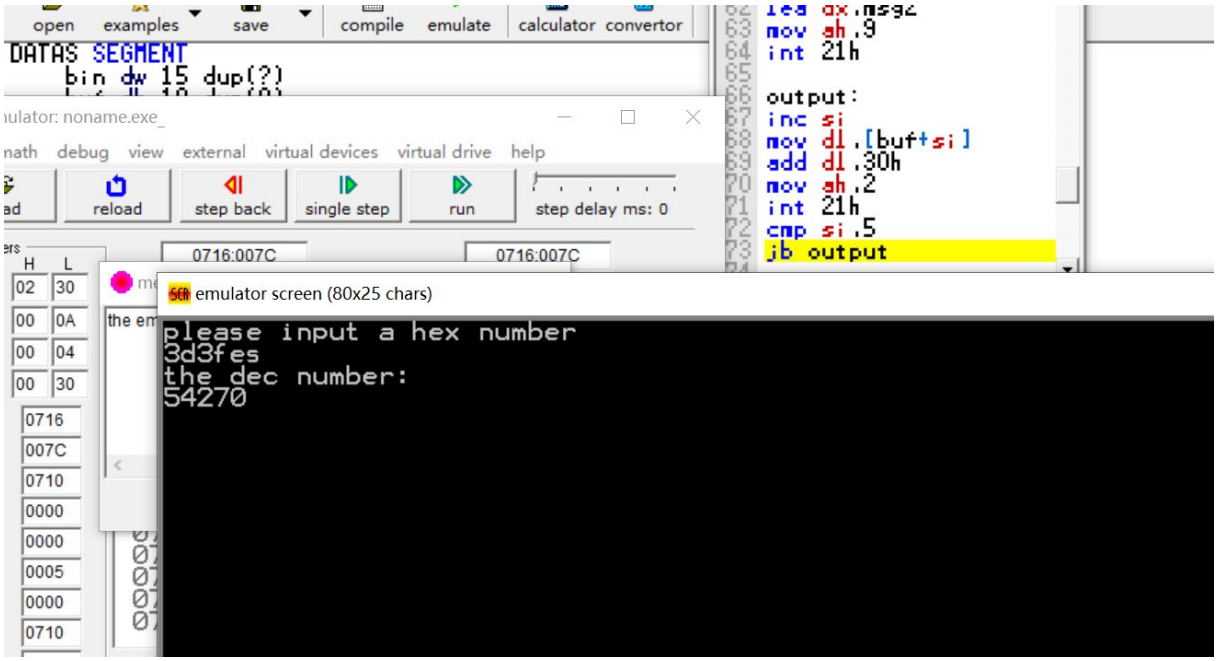




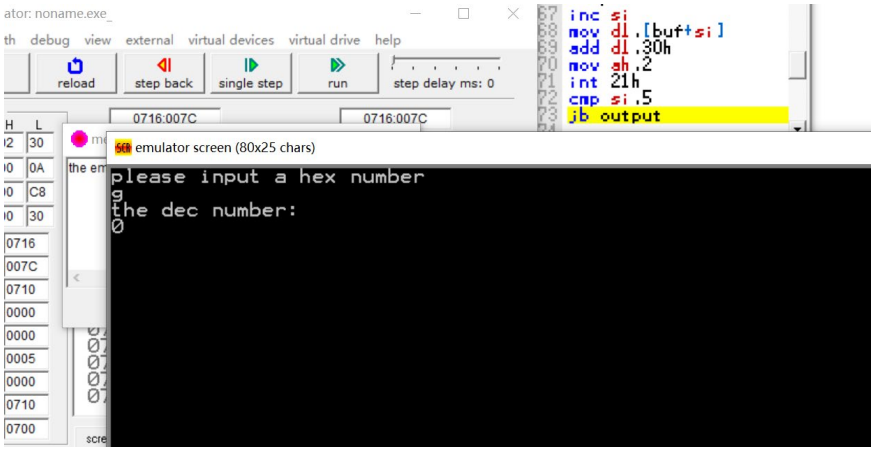
②a~f：



2.输入是极限 5 个字符时：



3.输入非 1~9、非 a~f 的时候

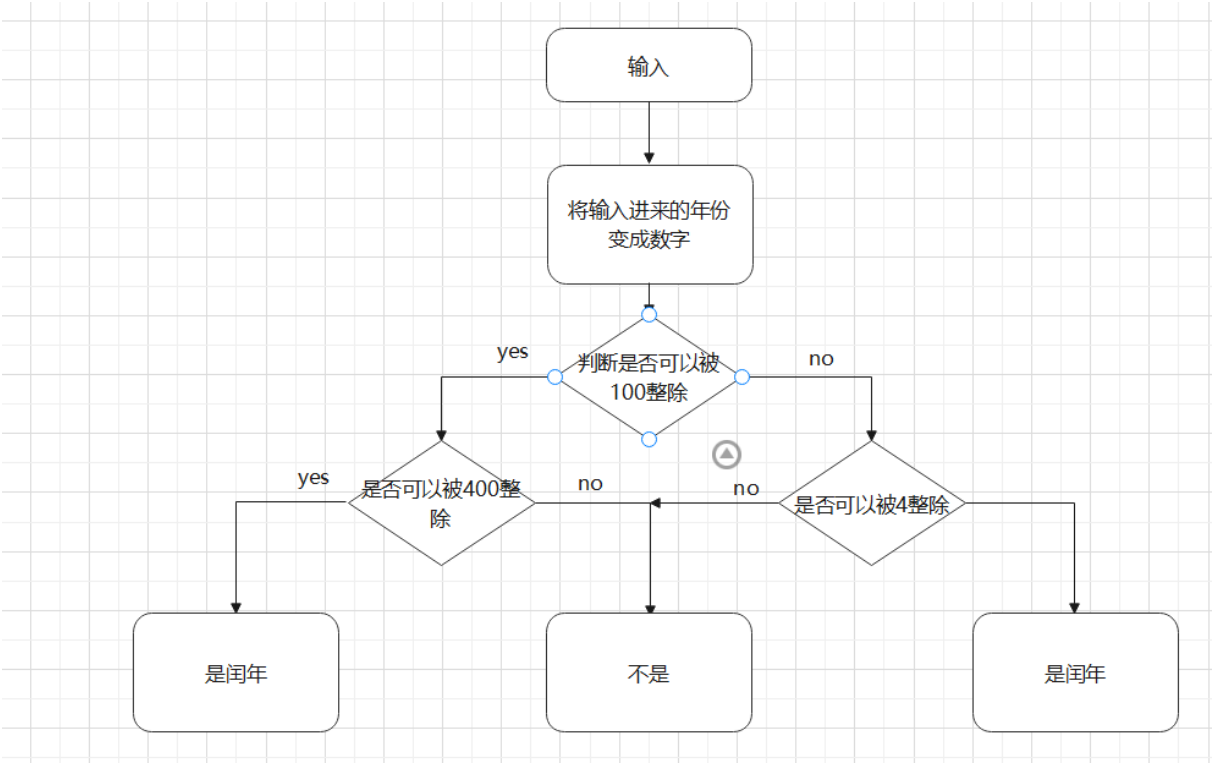


【实验心得】

这个十六进制转换成为十进制的程序感觉是上一个程序的拓展版，也有判断 ASCII 码的字符也有输入输出，所以还是很有趣的。实验 2.2 的时候也是编写程序，当时我自己写出来之后可能是有点兴奋也是懒了，没有上网再查一下人家的程序进行一个对照，改进一下，这回我结合自己和网上的思路改进了这个程序，看来之后还是要和他人多多学习才好

【第二个实验】欣赏判断年份代码

（1）实验流程图：



（2）实验源代码（粘贴源代码）：

```
;定义数据段，都是为了声明空间的
data segment
    ;输入提醒
    infon db 0dh,0ah,'Please input a year:$'
    ;判断输入的年份是闰年
    Y db 0dh,0ah,'Wow! This is a leap year!$'
    ;判断输入的年份不是闰年
    N db 0dh,0ah,'sorry, this is not a leap year!$'
    w dw 0
    ;其中的 buf 是一个缓冲区，被定义为字节型数据，
    buf db 8
        db 0
        db 8 dup(0)
data ends

;定义一个 200bytes 的字节
stack segment stack
```

```
    db 200 dup(0)
stack ends

code segment
        assume ds:data,ss:stack,cs:code
    start:
        ;指定数据段
        mov ax,data
        mov ds,ax

        ;屏幕上显示指定的 string
        lea dx,infon
        mov ah,9
        int 21h

        ;int 21h 指令在键盘上输入年份字符串
        lea dx,buf
        mov ah,10
        int 21h

        ;获取实际长度
        mov cl, [buf+1]
        ;保证 cx 的值是[buf+1]对应字节的值
        mov ch,0

        ;获取字符串的首地址，详情看缓冲区的解释
        lea di,buf+2
        ;调用子程序，将输入字符串化为年份数字
        call datacate
        ;调用子程序，判断是否为闰年
        call ifyears
        ;jc 表示如果进位标志为 1，则跳转到 a1
        jc    a1
```

```
        ;否则输出的不是闰年信息
        lea dx,n
        mov ah,9
        int 21h
        jmp exit
    ;输出是闰年信息
    a1:
        lea dx,y
        mov ah,9
        int 21h
    ;程序结束段
    exit:
        mov ah,4ch
        int 21h

;通过 near 指名该子程序在主程序段内
datacate proc near
        ;进行 cx 备份
        push cx
        ;cx 自减 1
        dec cx
        ;将 buf 的首地址赋给 si
        lea si,buf+2
    ;循环，然后使得 si 直接指向最后一个字符
    tt1: inc si
        loop tt1
        ;恢复 cx
        pop cx

        ;将字符转换成为数字
        mov dh,30h
        ;向前进一位，这个是当作十进制来处理的？
        mov bl,10
        ;ax 当作对应位的权值
```

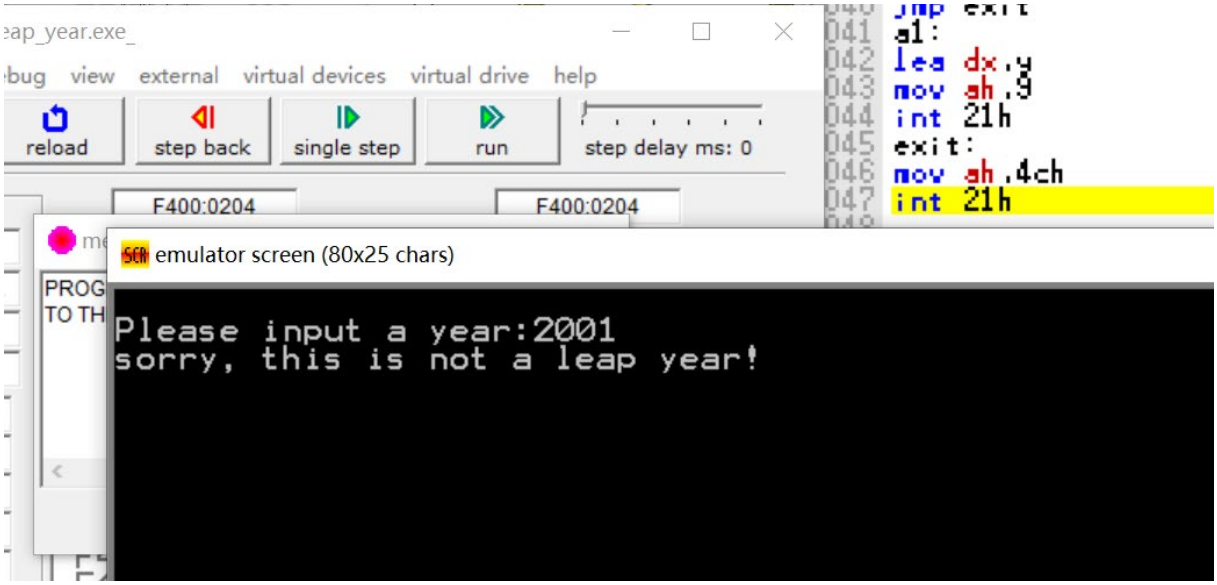
```
        mov ax,1
    l1:push ax
    push bx
    ;dx 是接受乘法运算结果的高位
    push dx
        ;将单个字符转换为对应的数字
        sub byte ptr [si],dh
        ;获取该位数字
        mov bl,byte ptr[si]
    mov   bh,0
      ;将该位乘以相应的权值，结果在 ax 中，因为年份不会超过两个字节表示
      mul   bx
      ;所以的结果相加得到年份
      add [w],ax
    ;都进行一顿的恢复
    pop dx
    pop bx
      pop ax
      ;权值乘以 10 使得 si 指向更高一位数字
      mul bl
      dec si
      loop l1
      ;子程序返回
      ret
datacate endp

;是判断是否为闰年的程序
ifyears proc near
      ;备份
      push bx
      push cx
      push dx
      ;获取年份的数据
      mov ax,[w]
```

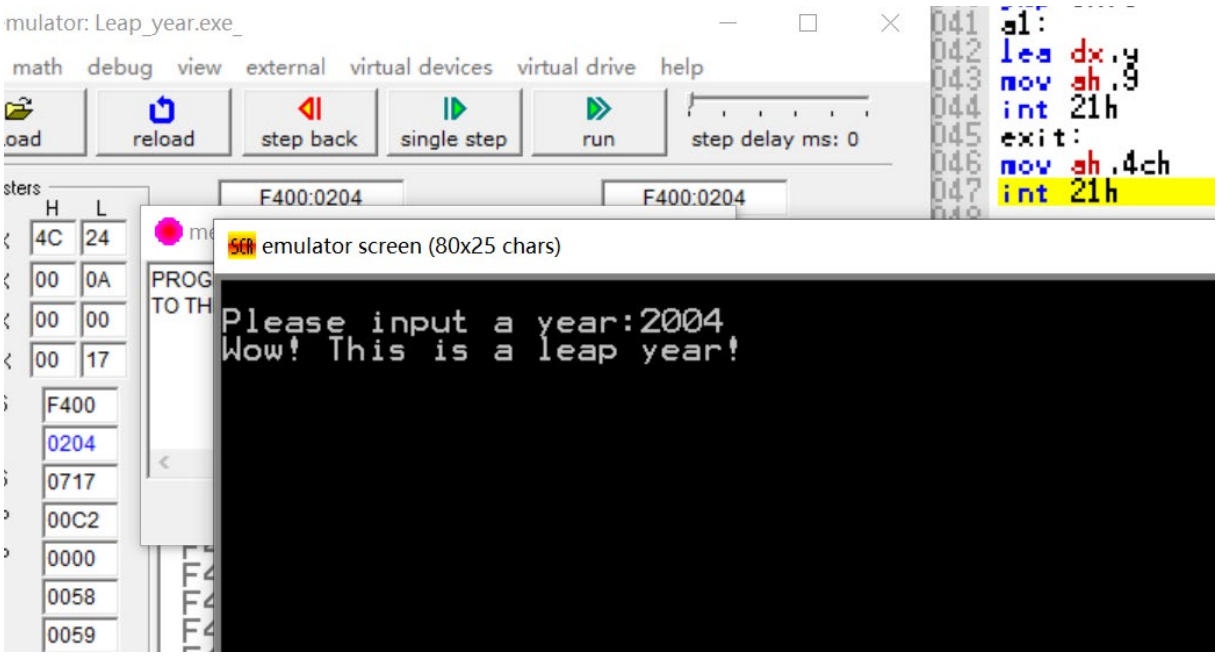
```
;将年份数据备份到 cx 中
mov cx,ax
;因为被除数要 32 字节，高位在 dx
mov dx,0

;这三行判断是否能被 100 整除
mov bx,100
div bx
cmp dx,0
;若不能则跳转到 lab1
jnz lab1
mov ax,cx
;判断是否可以被 400 整除
mov bx,400
div bx
cmp dx,0
;若是闰年则跳转的 lab2
jz lab2
;清除标记位
clc
jmp lab3

;判断是否可以被 4 整除
lab1:
    mov ax,cx
    mov dx,0
    mov bx,4
    div bx
    cmp dx,0
;可以被 4 整除所以跳转
    jz lab2
    clc
    jmp lab3
lab2:stc
```

```
        ;恢复寄存器
        lab3:
            pop dx
            pop cx
            pop bx
            ret
        ifyears endp
code ends
    end start
```

（3）实验代码、过程、相应结果（截图）并对实验进行说明和分析：

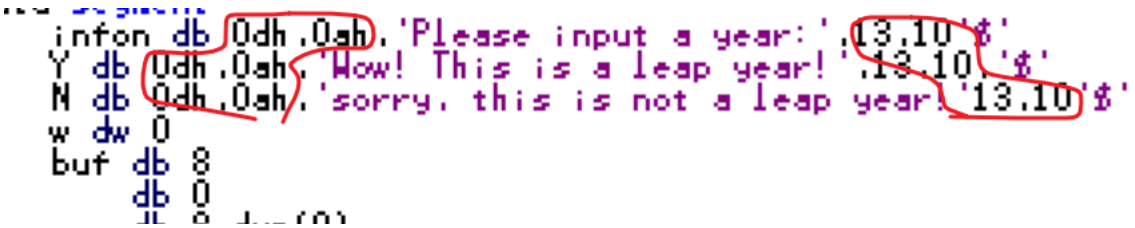
很简单的判断一个结果，这个实验程序的主要内容是理解程序





【实验心得】

1.有关显示提醒字符串：



我还用了之前的写法写 13 和 10 来表示回车和换行，老师这个是十六进制的写法

2. 有关汇编定义缓冲区的思考：

BUF DB 81

DB ?

DB 81 DUP（0）

在内存中申请一个缓冲区为83个字节，首地址给BUF,缓冲区的第一个字节内放的是81，表示申请的存放数据的缓冲区的字节数为81个，第二个字节“？”表示的是实际存放的字节个数（就是说，你放入2个字节的数据，“？”变成2，放10个字节的数据，变成10）；DB表示的是分配一个或多个字节；输入的数据从第三个字节开始存放，存放至第82个字节，第81个字节存放回车符（0DH）,0DH作为输入数据的结束。DUP(0)表示的是存放数据的81个字节初始值全为0，即为：81 0 0 0 0……（第82个字节）0 ODH。

解释的很详尽了，也让我理解这里 `lea di,buf+2 ;获取字符串首地址` 为什么是获取字符串的首地址了。

3.关于老师写的实验程序的思考：

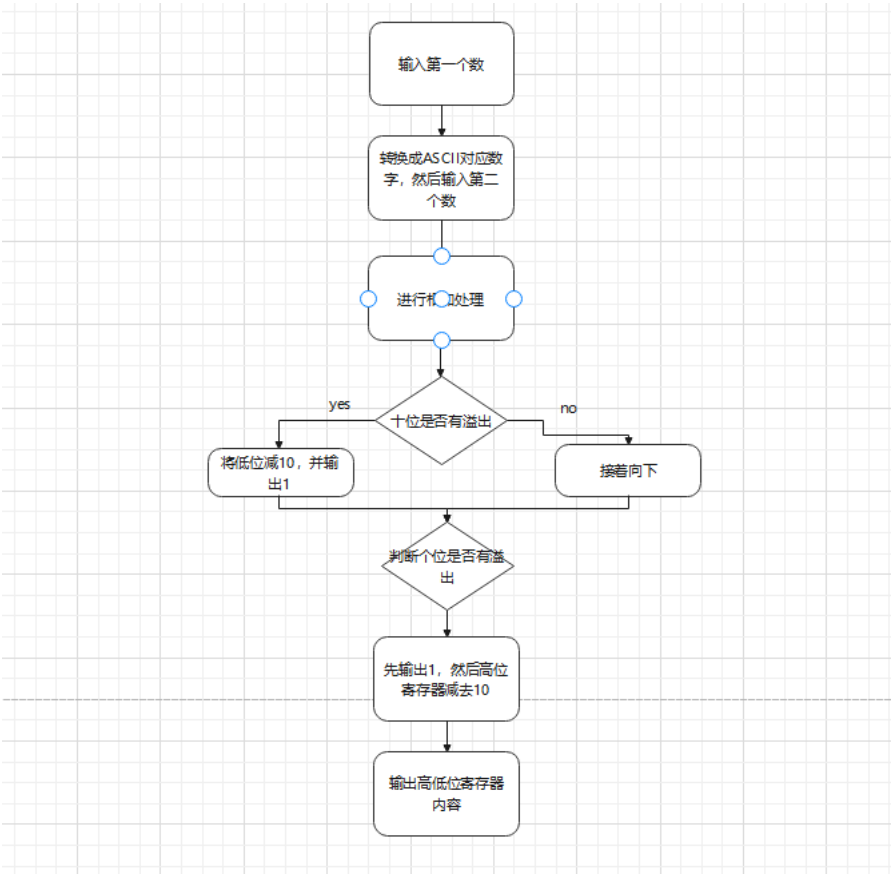
老师写的实验程序实属漂亮，开始我以为老师为什么有点对齐了，有的却没对齐，我在

“练打字”的时候就将它全对齐了，后来才发现老师的那个是每个程序的节点都会进行突出显示，而不是没有对齐

【第三个实验】实现汇编 3+5：

今天出去玩……人太多没回来，最后一个实验没时间写了，今天晚上就补上交个 update 下次不会这样了

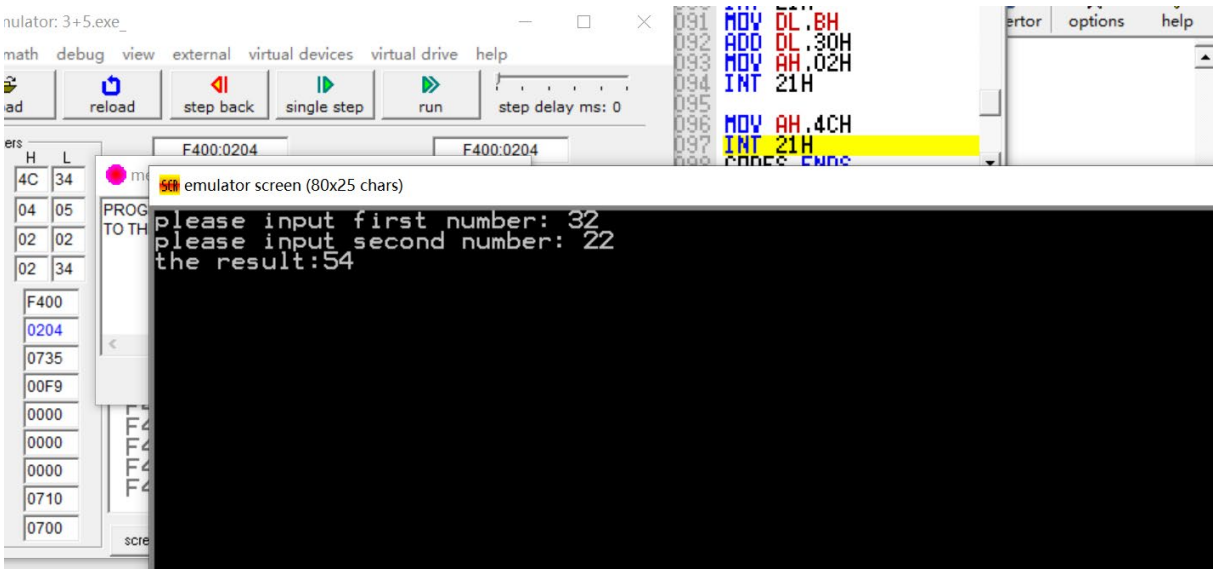
【实验流程图】



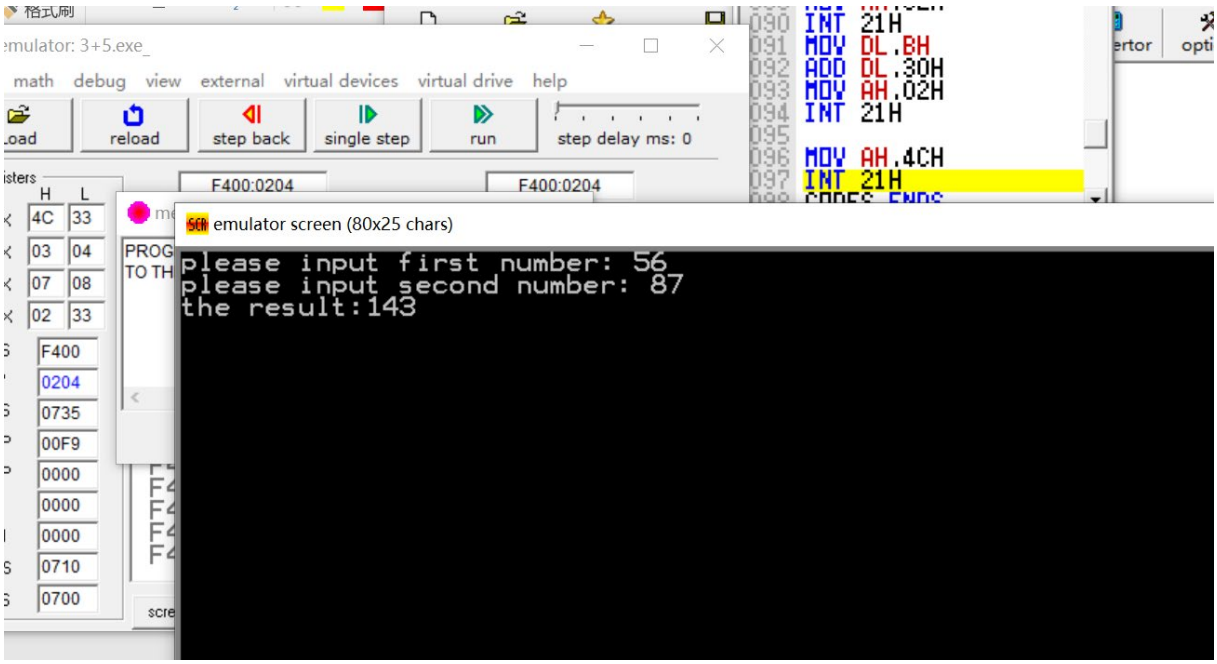
【实验结果】

1. 单个数相加

2.两位数相加（无溢出）



3.两位数相加（均溢出）



【实验代码】

```
DATAS SEGMENT
        ;定义了一个变量名为 data1    长度为 255 个字  的字符串
        data1 DB 255 dup('$')
        data2 DB 255 dup('$')
       ;模仿判断闰年的语句
        str1    DB 'please input first number: $ '
        str2    DB 'please input second number: $ '
        str3    DB 'the result:$'
        enter    DB    0DH,0Ah ,'$'
DATAS ENDS
STACK SEGMENT
                        ;定义了一个名为 lys 的堆栈
      lys DB 255 dup(0)
STACK ENDS
CODES SEGMENT
         ASSUME DS:DATAS,CS:CODES
START:
       MOV AX, DATAS
       MOV DS, AX

      LEA DX,str1
```

```
MOV AH,09H
INT 21H


;输入第一个两位数
LEA DX,data1
MOV AH,0AH
INT 21H
;输换行
LEA DX,enter
MOV AH,09H
INT 21H
;保存 data1 的个位和十位
MOV BL, [data1+2];十位
SUB BL,'0' ;这个是将 ASCII 转换为对应的数字
MOV BH, [data1+3];个位
SUB BH,'0'


LEA DX,str2
MOV AH,09H
INT 21H

;输入第二个两位数
LEA DX,data2
MOV AH,0AH
INT 21H
;输换行
LEA DX,enter
MOV AH,09H
INT 21H
;保存两位数的个位和十位
MOV CL,[data2+2]
SUB CL,'0'
```

```
    MOV CH,[data2+3]
    SUB CH,'0'

    ;开始相加
    ADD BL,CL
    ADD BH,CH

    LEA DX,str3
    MOV AH,09H
    INT 21H

    CMP BH,10
    JGE single_adding
tens_adding:
                    ;判断十进制的情况下是否有高位的溢出
    CMP BL,10
    JGE input_num
    JMP input_num2

single_adding:
    SUB BH,10
    ADD BL,1
    JMP tens_adding
;判断十位相加的时候是否有溢出
input_num:

    SUB BL,10
    MOV DL,1
    ADD DL,30H
    MOV AH,02H
    INT 21H
    JGE input_num2
;判断个位相加的时候是否有溢出
input_num2:
```

```
    MOV DL,BL
    ADD DL,30H
    MOV AH,02H
    INT 21H
    MOV DL,BH
    ADD DL,30H
    MOV AH,02H
    INT 21H

    MOV AH,4CH
    INT 21H
CODES ENDS
      END START
```

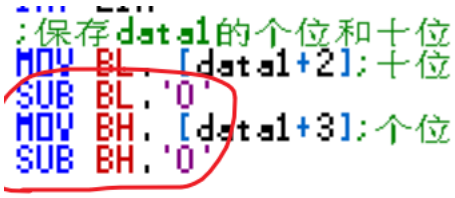
【实验心得】

1. int 21h 0A

Int 21h 的另外一种新用法，之前没有发现的

0A	键盘输入到缓冲区	DS:DX=缓冲区首地址 (DS:DX)=缓冲区最大字符数	(DS:DX+1)=实际输入的字符数

2. 在改进自己代码的时候



发现了这个，太优美了！老鸟啊，用字符‘0’来代替 30h 从而得到对应的数字

3.注意：

```
            JMP tens_adding
;判断十位相加的时候是否有溢出
input_num:

            SUB BL,10
            MOV DL,1
            ADD DL,30H
            MOV AH,02H
            INT 21H
            JGE input_num2
;判断个位相加的时候是否有溢出
input_num2:
            MOV DL,BL
            ADD DL,30H
            MOV AH,02H
            INT 21H
            MOV DL,BH
            ADD DL,30H
            MOV AH,02H
            INT 21H

            MOV AH,4CH
            INT 21H
CODES ENDS
```

要注意这个，因为数字在内存中是低地址存高位的

4.实验感悟：

以后完成任务一定要提前完成，补作业太痛苦了呜呜呜，质量不会很高而且心里还很累。