微机原理实验一

PB13206106
罗勇冠

一.实验目的

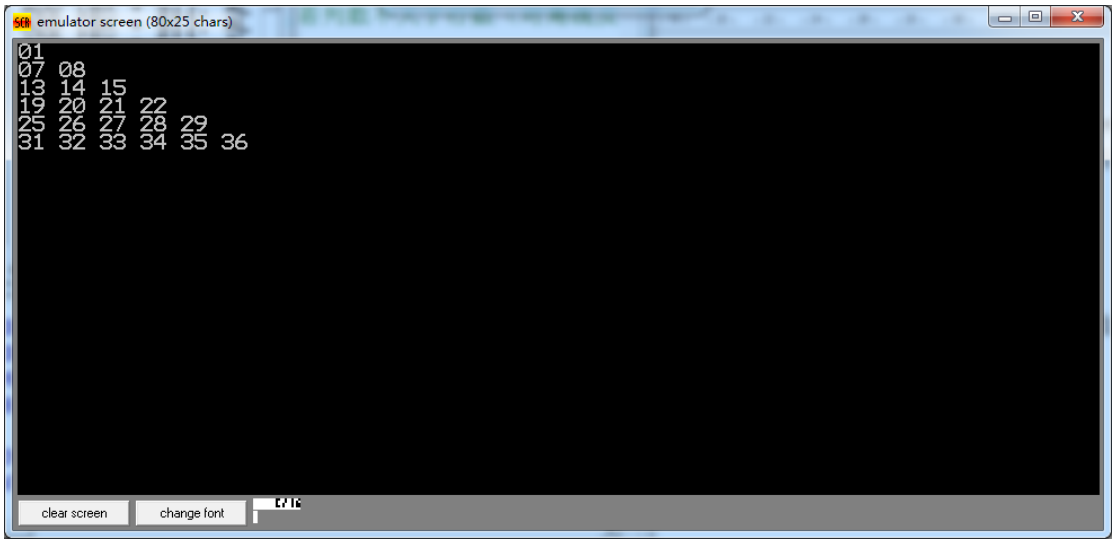
通过简单的汇编语言程序实验设计，学会 16 位汇编语言的编写与调试，熟悉 emu8086 实验环境，并学会使用其单步调试功能，观察汇编程序在执行时各寄存器的值以及每条汇编语句对应的内存地址。

二.实验内容

题 1：把 1~36 的自然数按行顺序存入一个 6*6 的二维数组中，然后打印出该数组的左下半三角。

三.实验结果

程序运行结果截图如下：



四.实验分析

编写实验代码前，又把课本上相关的几个完整段例子看了一遍，并在代码段前写下如下语句：

```
DATA SEGMENT
    ARRAY DB 36 DUP (0)        ;初始化数据组
DATA ENDS

STACK SEGMENT STACK
    DB 64 DUP (0)              ;定义栈空间
STACK ENDS
```

```
CODE SEGMENT
    ASSUME DS:DATA, CS:CODE, SS:STACK
```

由汇编语言基础知识可知，8088 汇编程序由代码段、数据段、标号和注释四个基本部分组成。代码段是包含指令的程序部分，这些指令完成传送数据、控制程序流程、实现算术运算功能等。数据段是汇编语言程序存放数据的部分。如以上定义了 ARRAY DB 36 DUP (0)即为保留了 36 个字节的不定值。汇编语言中的标号可用作变量名、段名或过程名，合法的标号由数字、字母和字符?、.、@_、和$组成，另外名称不能以数字打头，点号(.)只能用作标号的第一个字符。标号的长度不限，但只有前 31 个字符有效。除非标号用于连接汇编伪指令，否则后面必须跟一个冒号(:)。汇编语言的注释以;开头。ASSUME 是一个汇编程序伪指令，它向汇编程序通报 8088 段寄存器所指向的段。ASSUME DS:DATA, CS:CODE, SS:STACK 指定了数据段寄存器 DS 指向 DATA 段，代码段寄存器 CS 指向 CODE 段，堆栈段寄存器 SS 指向 STACK 段。

接下来，写了如下代码：

```
START:
    MOV AX, DATA                ;传输数据
    MOV DS, AX

    MOV DI, 0
    MOV AL, 1
    MOV CX, 36                  ;CX 循环变量设为 36
```

标号 START 指示了程序代码段的开始，而代码的前两句设置了数据段寄存器 DS 指向 DATA 段，由于该汇编文件编译后输出为 DOS EXE 程序，因此 DOS 自动设置代码段寄存器指向 CODE 段，而堆栈段寄存器指向程序上端的 STACK 段。然而，数据段寄存器的相应值必须由程序手工设置。

MOV DI,0 将 0 移入 DI 寄存器中，MOV AL, 1 将 1 移入 AL 寄存器中，以备后用。

MOV CX, 36 作用是为后面的循环指定了次数为 36.

然后我们就来到了程序的第一个循环部分：

```
GNUM:                           ;通过循环填入数据 1~36
    MOV ARRAY[DI], AL           ;ARRAY[0] = 1
    ADD AL, 1
    ADD DI, 1
    LOOP GNUM
```

循环的第一句 MOV ARRAY[DI], AL 将之前存放在 AL 中的数据 1 存储到 ARRAY 偏移空间中，初始偏移量 DI 为 0，即数组的第一个数据为 1，接下来的语句各将 AL 和 DI 加 1，然后继续循环执行 MOV ARRAY[DI], AL 直至 CX 由 36 减小为 0，便可以将 1 到 36 这 36 个数据存放至之前 ARRAY DB 36 DUP (0)保留的 36 个字节不定值中。

继续往下，DI 和 BL 这两个寄存器赋 0 初始值。

```
    MOV DI, 0
    MOV BL, 0
```

L1 循环如下，为 CL 赋 0 初始值，并且 DI 进栈。

```
    L1:
    MOV CL, 0
    PUSH DI
```

L2 循环如下，

```
L2:
MOV AL, ARRAY[DI]          ;将数组中数字存入 AL 中
XOR AH, AH                 ;AH 清零

PUSH AX                    ;AX 进栈保护
PUSH BX                    ;BX 进栈保护

MOV BL, 10                 ;BL 为 10
DIV BL                     ;AX 内容除以 BL 内容，AL 中存放得到的无符号的商，余数在 AH 中
MOV BH, AH                 ;余数存放至 BH,即个位数

CMP AL, 0                  ;相除的商和 0 比较
JA LL                      ;商大于 0 则跳转，即 AX 中数字为两位数
```

下面的 LL 循环为若十位数不为 0 所跳转到的分支：

```
LL:                        ;大于 10，输出十位
ADD AL, 30H                ;十位数转换成 ASCII 码
MOV DL, AL                 ;AL 中的十位
MOV AH, 2
INT 21H
```

以下语句输出个位和数字间的空格：

```
MOV AL, BH                 ;个位数移入 AL 中
ADD AL, 30H                ;个位数转换成 ASCII 码
MOV DL, AL
MOV AH, 2
INT 21H                    ;输出个位

MOV DL, 20H
MOV AH, 2
INT 21H                    ;输出数间空格

POP BX                     ;弹出 BX
POP AX                     ;弹出 AX,即数组数据

ADD CL, 1                  ;CL 记录列数，加 1
ADD DI, 1
```

以下语句为控制输出矩阵左下角的部分：

```
CMP CL, BL
JBE L2
```

再往下，当列数大于行数，即到了不用输出数字的部分，输出回车及换行，并控制好循环继续调回 L1 执行：

```
POP DI
```

```
MOV DL, 10
MOV AH, 2
INT 21H                    ;换行

MOV DL, 13
MOV AH, 2
INT 21H                    ;回车

ADD BL, 1                  ;增加行数
ADD DI, 6
CMP DI, 30
JBE L1                     ;若未满六行则继续处理

MOV AH, 4CH                ;若无错误则返回
INT 21H                    ;返回 DOS
```

最后程序结束，CODE ENDS 表示代码段结束，END START 指示代码段标号段结束：

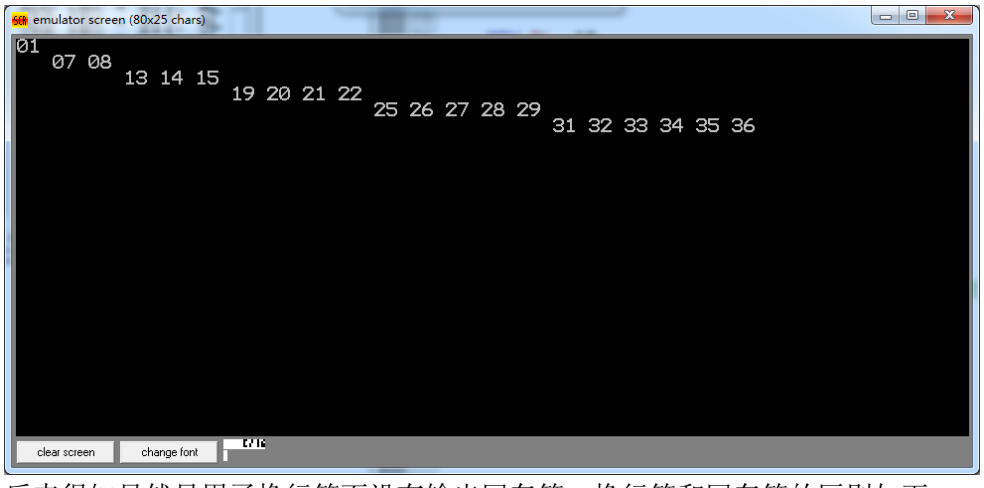
```
CODE ENDS
END START
```

五.意见建议

（1）关于实验过程中遇到的问题及解决方式：

1.由于用的操作系统为 Win7 64 位，运行 emu8086 时一开始编译结束后显示无法识别语句 ARRAY DB 36 DUP (?)，经过查找解决方案时了解到在 Win7 中，emu8086 不可以用 dup(?)来分配未初始化的单元，解决方案是把?改成 0 或者默认的数据就可以编译通过了，该问题可能是 emu8086 编译器的 bug。

2.输出时，一开始未了解回车和换行符的区别，导致第一次输出结果时，显示如下：



后来得知显然只用了换行符而没有输出回车符，换行符和回车符的区别如下：

符号	ASCII 码	意义
\n	10	换行 NL
\r	13	回车 CR

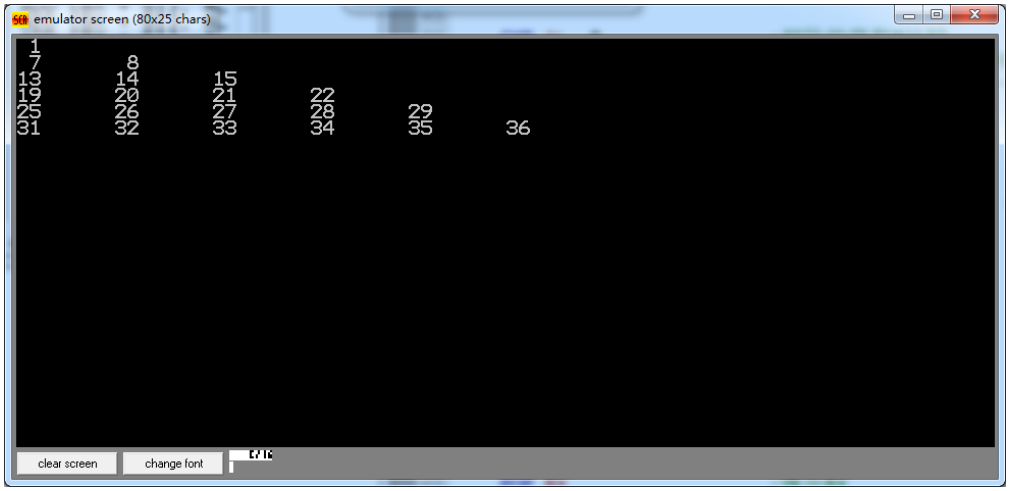
解决方案是在输出换行符的代码后面增加一段代码再输出回车符：

```
MOV DL, 13
```

```
    MOV AH, 2
    INT 21H                    ;回车
```

其中 13 为'\r'回车符的 ASCII 码的十进制数字。

（2）实验扩展，把个位数前的 0 去掉，并把数字与数字之间的空格符换成水平制表符，打印输出的结果如下：



修改方式为将除运算中的商，即个位数 0 改成空格符输出，把数字间的空格 ASCII 码 20H 改为水平制表符 ASCII 码 09H。

六.源代码

```
;实验一：把 1~36 的自然数按行顺序存入一个 6*6 的二维数组中，
;然后打印出该数组的左下半三角。

DATA SEGMENT
    ARRAY DB 36 DUP (0)        ;初始化数据组
DATA ENDS

STACK SEGMENT STACK
    DB 64 DUP (0)              ;定义栈空间
STACK ENDS

CODE SEGMENT
    ASSUME DS:DATA, CS:CODE, SS:STACK

    START:
    MOV AX, DATA               ;传输数据
    MOV DS, AX

    MOV DI, 0
    MOV AL, 1
    MOV CX, 36                 ;CX 循环变量设为 36
    GNUM:                      ;通过循环填入数据 1~36
    MOV ARRAY[DI], AL
```

```
    ADD AL, 1
    ADD DI, 1
    LOOP GNUM

    MOV DI, 0
    MOV BL, 0

    L1:
    MOV CL, 0
    PUSH DI

    L2:
    MOV AL, ARRAY[DI]          ;将数组中数字存入 AL 中
    XOR AH, AH                 ;AH 清零

    PUSH AX                    ;AX 进栈保护
    PUSH BX                    ;BX 进栈保护

    MOV BL, 10                 ;BL 为 10
    DIV BL                     ;AX 内容除以 BL 内容，AL 中存放得到的无符号的商，余数在
AH 中
    MOV BH, AH                 ;余数存放至 BH,即个位数

    CMP AL, 0                  ;相除的商和 0 比较
    JA LL                      ;商大于 0 则跳转，即 AX 中数字为两位数
    ;SUB AL, 10H               ;商为 0，即 AX 中数字为一位数，先减去 10H 结果为-10H
    LL:                        ;大于 10，输出十位
    ADD AL, 30H                ;十位数转换成 ASCII 码
    MOV DL, AL                 ;AL 中的十位
    MOV AH, 2
    INT 21H                    ;若大于 10 则输出十位，小于 10 则输出空格

    MOV AL, BH                 ;个位数移入 AL 中
    ADD AL, 30H                ;个位数转换成 ASCII 码
    MOV DL, AL
    MOV AH, 2
    INT 21H                    ;输出个位

    MOV DL, 20H
    MOV AH, 2
    INT 21H                    ;输出数间空格

    POP BX                     ;弹出 BX
    POP AX                     ;弹出 AX,即数组数据
```

```
    ADD CL, 1                ;CL 记录列数，加 1
    ADD DI, 1

    CMP CL, BL
    JBE L2                   ;若列数不大于行数（对角线及以左）则继续处理

    POP DI

    MOV DL, 10
    MOV AH, 2
    INT 21H                  ;换行

    MOV DL, 13
    MOV AH, 2
    INT 21H                  ;回车

    ADD BL, 1                ;增加行数
    ADD DI, 6
    CMP DI, 30
    JBE L1                   ;若未满六行则继续处理

    MOV AH, 4CH              ;若无错误则返回
    INT 21H                  ;返回 DOS

CODE ENDS
END START
```