



山东大学

## 信息科学与工程学院

2019 – 2020 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 2.2

---

专 业 班 级 通信工程 二班

学 生 学 号 201800121050

学 生 姓 名 孟麟芝

实 验 时 间 2020 年 2 月 26 日

# 实验报告

## 【实验目的】

1. 掌握控制转移类指令的使用
2. 掌握分支结构程序的编写
3. 掌握简单的循环结构程序的编写
4. 掌握一些简单中断程序的使用

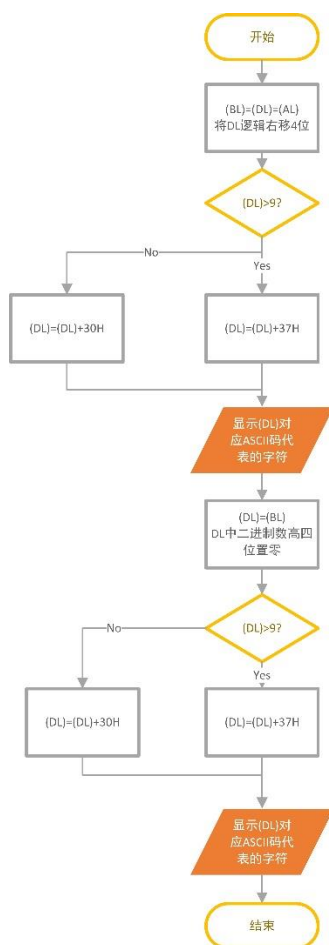
## 【实验要求】

1. 编写一个程序，将 AL 寄存器中的两位十六进制数显示出来
2. 编写一个程序，判别键盘上输入的字符；若是 1-9 字符，则显示之；若为 A-Z 或 a-z 字符，均显示 'c'；若是回车字符(其 ASCII 码为 0DH)，则自动结束程序，若为其它字符则不显示，循环等待新的字符输入。
3. 完成两个设计，截图显示实验结果，并贴出程序流程图

## 【实验具体内容】

### 【第一个实验】

#### 1.程序流程图：



## 2.程序源代码:

CODE SEGMENT

ASSUME CS:CODE ;这里使用伪指令定义了一个段为CODE, assume使之与CS段相关联

START:MOV AL, 3EH ;start设置了程序的入口, 防止数据与程序同时存在一个段内时出现错误

MOV BL, AL

MOV DL, AL ;这两步对AL的存放可以实现将两位十六进制数的处理分开

MOV CL, 4 ;把4赋给CL

SHR DL, CL ;将DL逻辑右移(CL)=4位(即不考虑符号), 右移后第四位放入CF, 高四位补0

CMP DL, 9 ;使用cmp指令, 通过结果不同对标志寄存器改变不同实现分支结构

JBE NEXT1 ;若(DL)<=9则直接进入NEXT1, 配合后面加30H, 可对应'0'~'9'的ASCII码

ADD DL, 7 ;若(DL)>9则DL加7, 则DL内为10D~15D时, 配合后面加30H, 已为65D~70D, 对应'A'~'F'的ASCII码

NEXT1:ADD DL, 30H ;如上所述, 便于对应ASCII码

MOV AH, 2

INT 21H ;这两步使用int指令, 显示AL中二进制数高四位 ASCII 码

MOV DL, BL ;把(AL)再次放入DL中操作

AND DL, 0FH ;进行逻辑与运算, 将DL中二进制数高四位置零, 第四位保持不变

CMP DL, 9

JBE NEXT2 ;与CMP配合, 当(DL)<=9进入NEXT2, 与下面四行程序构成对低位的显示输出, 与上面对高位的显示原理完全相同

ADD DL, 7

NEXT2:ADD DL, 30H

MOV AH, 2

INT 21H ;显示低位 ASCII 码

MOV AH, 4CH ;

INT 21H ;

CODE ENDS ;返回DOS

END START

## 3.逐步调试验证

```

AX=FFFF BX=0000 CX=0031 DX=0000 SP=0000 BP=0000 SI=0000 DI=0000
DS=075A ES=075A SS=0769 CS=076A IP=0000  NU UP EI PL NZ NA PO NC
076A:0000 B03E          MOV     AL,3E ①
-t

AX=FF3E BX=0000 CX=0031 DX=0000 SP=0000 BP=0000 SI=0000 DI=0000
DS=075A ES=075A SS=0769 CS=076A IP=0002  NU UP EI PL NZ NA PO NC
076A:0002 8AD8          MOV     BL,AL ②
-t

AX=FF3E BX=003E CX=0031 DX=0000 SP=0000 BP=0000 SI=0000 DI=0000
DS=075A ES=075A SS=0769 CS=076A IP=0004  NU UP EI PL NZ NA PO NC
076A:0004 8AD0          MOV     DL,AL ③
-t

AX=FF3E BX=003E CX=0031 DX=003E SP=0000 BP=0000 SI=0000 DI=0000
DS=075A ES=075A SS=0769 CS=076A IP=0006  NU UP EI PL NZ NA PO NC
076A:0006 B104          MOV     CL,04
    
```

经过前三步，预设好的(AL)=3EH 已被放入 BL,DL

```

AX=FF3E BX=003E CX=0031 DX=003E SP=0000 BP=0000 SI=0000 DI=0000
DS=075A ES=075A SS=0769 CS=076A IP=0006  NU UP EI PL NZ NA PO NC
076A:0006 B104          MOV     CL,04 ①
-t

AX=FF3E BX=003E CX=0004 DX=003E SP=0000 BP=0000 SI=0000 DI=0000
DS=075A ES=075A SS=0769 CS=076A IP=0008  NU UP EI PL NZ NA PO NC
076A:0008 D2EA          SHR     DL,CL ②
-t

AX=FF3E BX=003E CX=0004 DX=0003 SP=0000 BP=0000 SI=0000 DI=0000
DS=075A ES=075A SS=0769 CS=076A IP=000A  NU UP EI PL NZ AC PE CY
076A:000A 80FA09       CMP     DL,09 ①
-t

AX=FF3E BX=003E CX=0004 DX=0003 SP=0000 BP=0000 SI=0000 DI=0000
DS=075A ES=075A SS=0769 CS=076A IP=000D  NU UP EI NG NZ AC PE CY
076A:000D 7603         JBE     0012 ②
-t
    
```

这两步实现了对 DL 中二进制数的逻辑右移四位，保留了(DL)的高位

```

AX=FF3E BX=003E CX=0004 DX=0003 SP=0000 BP=0000 SI=0000 DI=0000
DS=075A ES=075A SS=0769 CS=076A IP=000A  NU UP EI PL NZ AC PE CY
076A:000A 80FA09       CMP     DL,09 ①
-t

AX=FF3E BX=003E CX=0004 DX=0003 SP=0000 BP=0000 SI=0000 DI=0000
DS=075A ES=075A SS=0769 CS=076A IP=000D  NU UP EI NG NZ AC PE CY
076A:000D 7603         JBE     0012 ②
-t
    
```

这两步对 DL 中的数判断是否大于 9，以方便匹配不同的 ASCII 码，0012 对应的是源程序中 NEXT1 第一条指令的偏移地址

```

AX=FF3E BX=003E CX=0004 DX=0003 SP=0000 BP=0000 SI=0000 DI=0000
DS=075A ES=075A SS=0769 CS=076A IP=0012  NU UP EI NG NZ AC PE CY
076A:0012 80C230       ADD     DL,30 ①
-t

AX=FF3E BX=003E CX=0004 DX=0033 SP=0000 BP=0000 SI=0000 DI=0000
DS=075A ES=075A SS=0769 CS=076A IP=0015  NU UP EI PL NZ NA PE NC
076A:0015 B402         MOV     AH,02 ②
-t

AX=023E BX=003E CX=0004 DX=0033 SP=0000 BP=0000 SI=0000 DI=0000
DS=075A ES=075A SS=0769 CS=076A IP=0017  NU UP EI PL NZ NA PE NC
076A:0017 CD21         INT     21 ③
    
```

上面判断 DL 高位为小于 9 的数（3），故直接加 30H，对应了'3'的 ASCII 码

```

AX=023E BX=003E CX=0004 DX=0033 SP=FFFA BP=0000 SI=0000 DI=0000
DS=075A ES=075A SS=0769 CS=F000 IP=14A0  NU UP DI PL NZ NA PE NC
F000:14A0 FB          STI 1
-t

AX=023E BX=003E CX=0004 DX=0033 SP=FFFA BP=0000 SI=0000 DI=0000
DS=075A ES=075A SS=0769 CS=F000 IP=14A1  NU UP EI PL NZ NA PE NC
F000:14A1 FE38      ???  [BX+SI] 2          DS:003E=00
-t
3

AX=0233 BX=003E CX=0004 DX=0033 SP=FFFA BP=0000 SI=0000 DI=0000
DS=075A ES=075A SS=0769 CS=F000 IP=14A5  NU UP EI PL NZ NA PE NC
F000:14A5 CF          IRET 3
-t
    
```

在这里，int21 用-t 命令调试是会出现问题的，如图所示，第一步 STI 允许中断，第二步则是未知指令，第三步又进行了返回，虽然程序也可以继续向下运行，不过已经导致了未知的操作。这里应该使用-p 命令调试

```

AX=023E BX=003E CX=0004 DX=0033 SP=0000 BP=0000 SI=0000 DI=0000
DS=075A ES=075A SS=0769 CS=076A IP=0017  NU UP EI PL NZ NA PE NC
076A:0017 CD21          INT 21
-p
3
    
```

这样就看到了我们想要的结果

```

AX=0233 BX=003E CX=0004 DX=0033 SP=0000 BP=0000 SI=0000 DI=0000
DS=075A ES=075A SS=0769 CS=076A IP=0019  NU UP EI PL NZ NA PE NC
076A:0019 8AD3          MOV DL,BL 1
-t

AX=0233 BX=003E CX=0004 DX=003E SP=0000 BP=0000 SI=0000 DI=0000
DS=075A ES=075A SS=0769 CS=076A IP=001B  NU UP EI PL NZ NA PE NC
076A:001B 80E20F        AND DL,0F 2
-t

AX=0233 BX=003E CX=0004 DX=000E SP=0000 BP=0000 SI=0000 DI=0000
DS=075A ES=075A SS=0769 CS=076A IP=001E  NU UP EI PL NZ NA PO NC
076A:001E 80FA09        CMP DL,09
    
```

再将(BL)=3EH 放入 DL，且将 DL 四进制数高四位置 0，可见保留了低位

```

AX=0233 BX=003E CX=0004 DX=000E SP=0000 BP=0000 SI=0000 DI=0000
DS=075A ES=075A SS=0769 CS=076A IP=001E  NU UP EI PL NZ NA PO NC
076A:001E 80FA09          CMP     DL,09 ①
-t

AX=0233 BX=003E CX=0004 DX=000E SP=0000 BP=0000 SI=0000 DI=0000
DS=075A ES=075A SS=0769 CS=076A IP=0021  NU UP EI PL NZ NA PE NC
076A:0021 7603          JBE     0026 ②
-t

AX=0233 BX=003E CX=0004 DX=000E SP=0000 BP=0000 SI=0000 DI=0000
DS=075A ES=075A SS=0769 CS=076A IP=0023  NU UP EI PL NZ NA PE NC
076A:0023 80C207          ADD     DL,07 ③
-t

AX=0233 BX=003E CX=0004 DX=0015 SP=0000 BP=0000 SI=0000 DI=0000
DS=075A ES=075A SS=0769 CS=076A IP=0026  NU UP EI PL NZ AC PO NC
076A:0026 80C230          ADD     DL,30 ④

```

比较得(DL)中得数大于 10（为 EH），三四步加 37H 使之对应'E'得 ASCII 码

```

AX=0233 BX=003E CX=0004 DX=0045 SP=0000 BP=0000 SI=0000 DI=0000
DS=075A ES=075A SS=0769 CS=076A IP=0029  NU UP EI PL NZ NA PO NC
076A:0029 B402          MOV     AH,02 ①
-t

AX=0233 BX=003E CX=0004 DX=0045 SP=0000 BP=0000 SI=0000 DI=0000
DS=075A ES=075A SS=0769 CS=076A IP=002B  NU UP EI PL NZ NA PO NC
076A:002B CD21          INT     21 ②
-p
E

```

再次利用 int21 中断程序输出，还是要注意使用-p 指令

```

AX=0245 BX=003E CX=0004 DX=0045 SP=0000 BP=0000 SI=0000 DI=0000
DS=075A ES=075A SS=0769 CS=076A IP=002D  NU UP EI PL NZ NA PO NC
076A:002D B44C          MOV     AH,4C
-t

AX=4C45 BX=003E CX=0004 DX=0045 SP=0000 BP=0000 SI=0000 DI=0000
DS=075A ES=075A SS=0769 CS=076A IP=002F  NU UP EI PL NZ NA PO NC
076A:002F CD21          INT     21
-p

Program terminated normally

```

调试完毕，可以直接运行程序查看一下效果

```

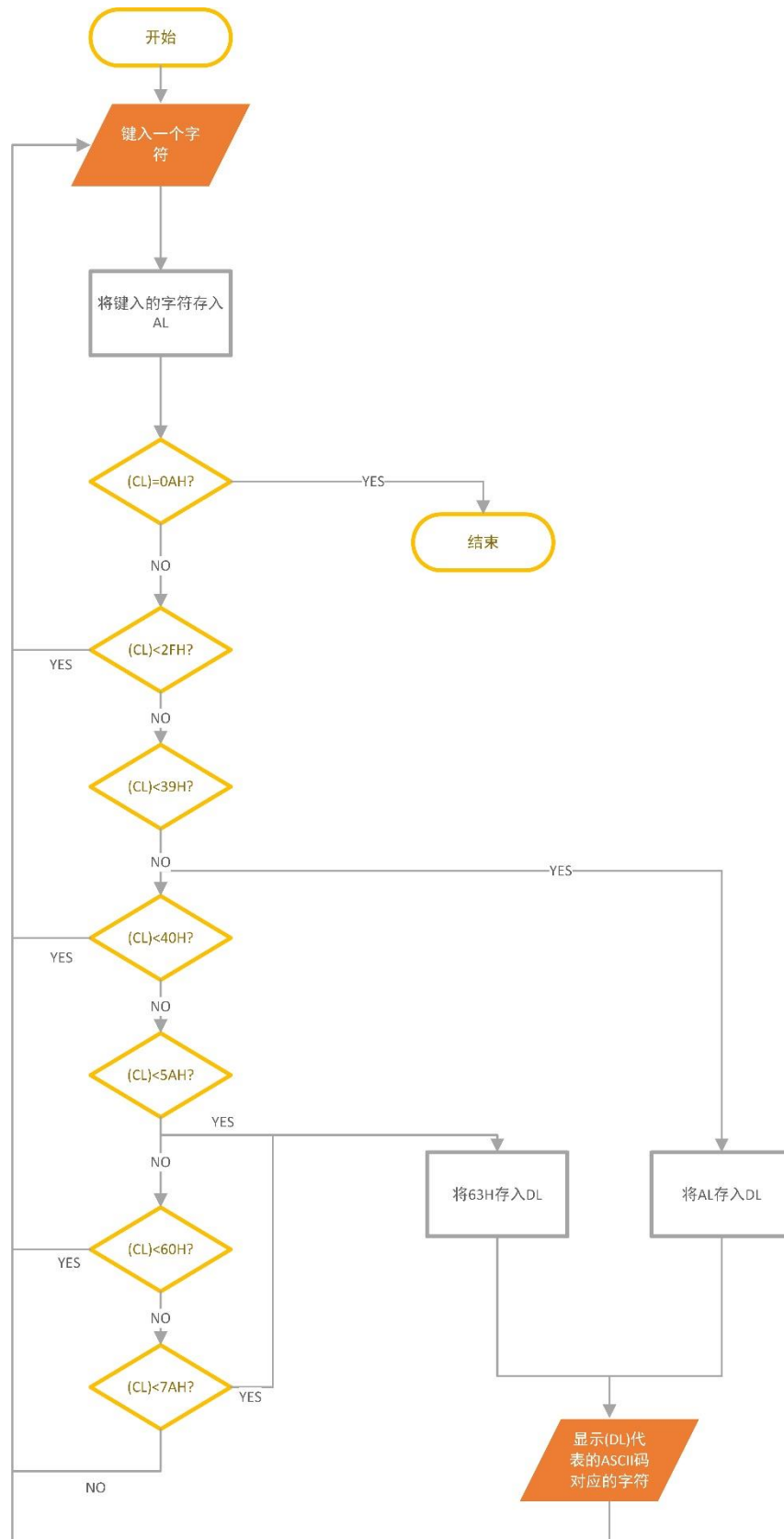
C:\>PI.EXE
3E
C:\>

```

也是符合预期的

## 【第二个实验】

### 1.程序流程图:



## 2.程序源代码

```
CODE SEGMENT
    ASSUME CS:CODE
START:RESTART:
    MOV AH, 07H
    INT 21H
    CMP AL, 0DH
    JE ENDD;回车则结束程序

    CMP AL, 2FH
    JBE RESTART;除去回车，ASCII码在0~47则重新开始
    CMP AL, 39H
    JBE NUM;ASCII码在48~57，进入数字处理块
    CMP AL, 40H
    JBE RESTART;ASCII码在58~64则重新开始
    CMP AL, 5AH
    JBE ALPHA;ASCII码在65~90，进入字母处理块
    CMP AL, 60H
    JBE RESTART;ASCII码在91~96则重新开始
    CMP AL, 7AH
    JBE ALPHA;ASCII码在97~122，进入字母处理块
    JMP RESTART;ASCII码在123~126则重新开始

NUM:MOV DL, AL
    JMP DISPLAY
;实现对输入数字的显示

ALPHA:MOV DL, 63H
    JMP DISPLAY
;63H对应'c'的ASCII码，实现输入字母显示'c'

DISPLAY:MOV AH, 02H
    INT 21H
    JMP RESTART
;使用int指令输出(DL)对应字符，输出结束后重新开始

ENDD:MOV AH, 4CH
    INT 21H
CODE ENDS
END START
```

## 3.程序运行验证

(1) 对 p2.asm 进行编译链接



```
C:\>masm
Microsoft (R) Macro Assembler Version 5.00
Copyright (C) Microsoft Corp 1981-1985, 1987. All rights reserved.

Source filename [.ASM]: p2.asm
Object filename [p2.OBJ]:
Source listing [NUL.LST]:
Cross-reference [NUL.CRF]:

51770 + 464774 Bytes symbol space free


0 Warning Errors
0 Severe Errors
```

```
C:\>link p2.obj

Microsoft (R) Overlay Linker Version 3.60
Copyright (C) Microsoft Corp 1983-1987. All rights reserved.

Run File [P2.EXE]:
List File [NUL.MAP]:
Libraries [.LIB]:
LINK : warning L4021: no stack segment
```

无栈段的提醒是通过常规对程序的检验发出的，可以忽略

 P2.EXE

可见已生成可执行文件

## (2) 程序运行验证

```
C:\>p2
1234567890cccccc
```

经过验证，输入数字直接显示，输入大小写字母均显示为‘c’，而输入其他字符，无变化

```
C:\>p2
1234567890cccccc
C:\>
```

敲击回车实现退出程序（要注意，CTRL+CR 对应的 ASCII 码为 0AH，二者有区别）

## 【实验心得】

1. CTRL+CR 对应的 ASCII 码为 0AH，CR 对应的为 0DH,二者有区别
2. 对于复杂程序的设计可以使用模块化的思想
3. 对 INT21H 要使用 -p 指令执行，-t 指令执行后会引发未知的操作
4. 出于对程序的常规检查，masm 编译器会对无栈段的代码发出 warning，若无需使用栈段则可以忽略
5. 要注意逻辑右移和算术右移的区别，逻辑右移直接在高位补 0，算术右移考虑了符号位，为负数时，右移后最高位为 1，正数则与逻辑右移无区别

## 6. INT21 程序使用时要注意好发生条件和操作结果

### 【思考】

1.DS 和 ES 有无存放偏移地址的寄存器？如有得话如何查看？

一、DS 为数据段，ES 为附加段，在一般情况下二者偏移地址 EA 可由以下几种方式给出

(1) 直接寻址

EA=[idata]

(2) 间接寻址

EA=[BX] EA=[SI] EA=[DI]

(3) 寄存器相对寻址

EA=[BX+idata] EA=[SI+idata] EA=[DI+idata]

(4) 基址变址寻址

EA=[BX+SI] EA=[BX+DI]

(5) 相对基址变址寻址

EA=[BX+SI+idata] EA=[BX+DI+idata]

二、涉及中断等汇编指令时，二者有区别

如：在 INT21 中，一些调用参数由 DS:DX 决定，并不能通过设置 ES 为 DATA segment，将调用参数改变成由 ES:DX 决定，除非修改 INT21 中断程序

2.下面程序中若用 JL 如何实现？程序复杂度有何区别？

求 AX 中数据的绝对值：

CMP AX, 0

JGE NONEG

NEG AX

NONEG: MOV RESULT, AX

答：用 JL 实现，应为

CMP AX, 0

JL YNEG

JMP S

YNEG: NEG AX

S:MOV RESULT,AX

可见，对负数处理步骤数无区别，对正数需要多一步才得到结果