

# 实验九 时钟实验

学号： 04022212

姓名： 钟 源

## 一、实验任务和实验结果

请写出**已通过验收**的各个实验任务的具体内容、调试通过的源程序（**加注释**）和实验结果。

实验结果请截图，并加以必要说明。

### 1.任务：

(1) 执行时钟程序时，屏幕上显示提示符“: ”，由键盘输入当前时、分和秒值，即 XX:

XX:XX√，随即显示时间并不停地计时。

(2) 当有键按下时，立即停止计时，返回 DOS。

### 附加任务：

(1) 在同一行的相同位置显示更新的计时时间，不换行；

(2) 输入时间初值时，会检查是否有错、提示错误信息，并可重新输入时间初值。

错误提示信息可以分两种：

1) 输入的时间初值是错误的字符，即不是数字和冒号；(format)

2) 输入的时间值是错误的，即“时”大于等于 24，“分”和“秒”大于等于 60。

(3) 延时一秒用 DOS 系统功能调用实现

### 2.源程序及注释：

```
DATA    SEGMENT;           定义数据段
TIME    DB  9;
        DB  ?;
        DB  9  DUP(?);
ERROR1   DB  2  DUP(?);
        DB  'Wrong Format';
ERROR2   DB  2  DUP(?);
```

```

        DB 'Wrong Time';
DATA    ENDS

STA     SEGMENT  STACK;
        DB 20 DUP(0);
STA     ENDS

CODE     SEGMENT ;           定义代码段
ASSUME   CS: CODE, DS:DATA, SS:STA;
START:

    MOV  AX, DATA
    MOV  DS, AX;             给 DS 附上初值
    MOV  ES, AX;             给 ES 附上初值
    MOV  DL, ':';             给 dl 赋值，输出冒号
    MOV  AH, 2;
    INT  21H;                 在屏幕上输出冒号
    LEA  DX, TIME;           提供缓存区地址
    MOV  AH, 0AH;
    INT  21H;                 输入
    MOV  BX, OFFSET TIME+2;   BX 取到数组的首地址
    CALL INPUTError;          判断输入格式是否正确          实现附加功能 2

RIG:
    CALL ASCtoBCD;            把缓存区 ASC 码的转换成 BCD 码
    CALL TIMEError;           判断时间符合实际          实现附加功能 2

AGAIN:
    CALL MDELAY;              调用延迟函数
    CALL TIMECHANGE;          调用时间加一秒函数
    CALL SHOW;                 调用显示时间函数
    PUSH DX;                   压栈保护 DX
    MOV  AH, 06H;
    MOV  DL, 0FFH;
    INT  21H;                 判断是否有按键输入
    POP  DX;                   弹出 DX
    JNZ  AAA;                 有按键输入则跳转
    JMP  AGAIN;               没有按键输入，则继续计时

AAA:
    MOV  AH, 4CH;             返回 dos，主程序结束
    INT  21H;

;判断输入错误的子程序
INPUTError PROC
    MOV  CL, 00H;             开始 CL 置零

    MOV  AL, [BX];            提取 BX 对应的值

```

CALL NUMerror;	判断是否是数字
INC BX;	判断下一字节
MOV AL, [BX];	提取 BX 对应的值
CALL NUMerror;	判断是否是数字
INC BX;	判断下一字节
MOV AL, [BX];	提取 BX 对应的值
CALL COLONerror;	判断是否是冒号
INC BX;	判断下一字节
MOV AL, [BX];	提取 BX 对应的值
CALL NUMerror;	判断是否是数字
INC BX;	判断下一字节
MOV AL, [BX];	提取 BX 对应的值
CALL NUMerror;	判断是否是数字
INC BX;	判断下一字节
MOV AL, [BX];	提取 BX 对应的值
CALL COLONerror;	判断是否是冒号
INC BX;	判断下一字节
MOV AL, [BX];	提取 BX 对应的值
CALL NUMerror;	判断是否是数字
INC BX;	判断下一字节
MOV AL, [BX];	提取 BX 对应的值
CALL NUMerror;	判断是否是数字
CMP CL, 00H;	CL=0 说明格式正确
JE RIG ;	正确跳转

error:

```

MOV DX,OFFSET ERROR1
MOV BX, OFFSET ERROR1
MOV AL, 0DH;  输出回车
MOV [BX], AL
INC BX
MOV AL,0AH;   输出换行
MOV [BX],AL
ADD BX,13D;   指针跳转
MOV AL,0AH;   输出换行
MOV [BX],AL;
INC BX

```

```

MOV AX, '$';    输出结尾标志
MOV [BX], AL

MOV AH, 9
INT 21H;        在屏幕显示 ERROR1
JMP START;      跳回开头
RET
INPUTError ENDP

;判断数字错误的子程序
NUMError PROC
CMP AL, 39H;
JA ERR1;
CMP AL, 30H;
JB ERR1;        以上都是比较 ASC 码，30，39 为数字在 ASC 码表中范围
RET
ERR1:
MOV CL, 01H;    其 ASC 码若不在数字范围，CL 置 1
RET
NUMError ENDP

;判断冒号错误的子程序
COLONError PROC
CMP AL, 3AH;    3AH 是“:”的 ASC 码
JNE ERR2;
RET
ERR2:
MOV CL, 01H;    ASC 码若不是规定值，CL 置 1
RET
COLONError ENDP

;ASC 变压缩 BCD 的子程序
ASCtoBCD PROC
MOV BX, OFFSET TIME+2;    指针指向数据区
CALL TRAN;                调用 TRAN 子程序
MOV CH, [BX];             把时针值给 CH
ADD BX, 3;
CALL TRAN;
MOV DH, [BX];             把分针值给 DH
ADD BX, 3;
CALL TRAN;
MOV DL, [BX];             把秒针值给 DL
RET
ASCtoBCD ENDP

```

;TRAN 子程序,辅助 ASCtoBCD

TRAN PROC

```
    MOV AL, [BX+1];
    AND AL, 0FH;    个位数字的 ASC 变 BCD
    MOV [BX+1], AL;
    MOV AL, [BX];
    SHL AL, 1;
    SHL AL, 1;
    SHL AL, 1;
    SHL AL, 1;    十位数字的 ASC 变 BCD
    ADD AL, [BX+1]; 变压缩 BCD, 用于后时间的加法计算
    MOV [BX], AL;
    RET
```

TRAN ENDP

;判断时间符合实际的子程序

TIMEError PROC

```
    MOV CL, 00H;
    CMP CH, 24H;    将“时”和 24 比较
    JAE DO;         和 24 等, 或>24 则跳转
    CMP DH, 60H;    将“分”和 60 比较
    JAE DO;         和 60 等, 或>60 则跳转
    CMP DL, 60H;    将“秒”和 60 比较
    JAE DO;         和 60 等, 或>60 则跳转
    JMP AGAIN
```

DO: MOV DX,OFFSET ERROR2

MOV BX, OFFSET ERROR2

MOV AL, 0DH; 输出回车

MOV [BX], AL

INC BX

MOV AL,0AH; 输出换行

MOV [BX],AL

ADD BX,11D; 指针跳转

MOV AL,0AH; 输出换行

MOV [BX],AL;

INC BX

MOV AX, '\$'; 输出结尾标志

MOV [BX], AL

MOV AH, 9

INT 21H; 在屏幕显示 ERROR2

JMP START; 跳回开头

RET

TIMEError ENDP

;延迟函数的定义（附加功能3）

```
MDELAY PROC
;压栈保护现场
PUSH AX;
PUSH BX;
PUSH CX;
PUSH DX;

MOV AH, 2CH ; 使用系统调用 INT 21H, AH=2CH 获取当前系统时间
INT 21H ; 【CH 小时,CL 分,DH 秒,DL 百分之一秒】

ADD DH, 1H ; 将当前秒数加 1
CMP DH, 3CH ; 与 60 比较, 如果不等于 60, 可以进行下一步
JNE NOTEQU
MOV DH, 00H ; 如果相等需要将 DH 置 0
NOTEQU:
MOV BL, DH ; 无论是否相等都会执行的程序写在跳转的语句中。
COMPARETIME:
MOV AH, 2CH
INT 21H
CMP BL, DH ; 比较备份的秒数和当前秒数是否相等, 相等则延时完成
JNE COMPARETIME ; 如果不相等, 继续等待
;弹出恢复现场
POP DX
POP CX
POP BX
POP AX
RET
MDELAY ENDP
```

;时间加一子程序

```
TIMECHANGE PROC
MOV AL, DL;
ADD AL, 1; 秒针加一
DAA; 压缩 BCD 码运算的重要流程
MOV DL, AL;
CMP AL, 60H; 秒针和 60 比较, 注! 这里一定是 60H, 不是 60
JNE DONE; 比 60 小, 直接 RET
MOV DL, 00H; >60, 进行进位调整
MOV AL, DH;
ADD AL, 1;
DAA;
MOV DH, AL;
```

```

    CMP    AL, 60H;           分钟和 60 比较
    JNE    DONE;             比 60 小，直接 RET
    MOV    DH, 00H;          >60，进行进位调整
    MOV    AL, CH;
    ADD    AL, 1;
    DAA;
    MOV    CH, AL;
    CMP    AL, 24H;          时针和 24 比较
    JNE    DONE;             比 24 小，直接 RET
    MOV    CH, 00H;          >24，进行进位调整
DONE:    RET
TIMECHANGE    ENDP

;SHOW 子程序
SHOW    PROC

    MOV    BX, OFFSET TIME    ; 将 TIME 的偏移地址加载到 BX 寄存器中
    ; 加载回车 '\r'
    MOV    AL, 0DH
    MOV    [BX], AL
    INC    BX
    ; 加载冒号 ':'
    MOV    AL, 3AH
    MOV    [BX], AL
    INC    BX
    ; 加载时钟值
    MOV    AL, CH              ; 将时钟值加载到 AL 寄存器中
    CALL    BCDtoASC           ; 调用 BCDtoASC 子程序将分钟值转换为 ASCII 码
    ADD    BX, 2               ; 将 BX 寄存器增加 2，指向 TIME 缓冲区的下一个位置
    ; 加载冒号 ':'
    MOV    AL, ':'
    MOV    [BX], AL
    INC    BX
    ; 加载分钟值
    MOV    AL, DH              ; 将分钟值加载到 AL 寄存器中
    CALL    BCDtoASC           ; 调用 BCDtoASC 子程序将分钟值转换为 ASCII 码
    ADD    BX, 2               ; 将 BX 寄存器增加 2，指向 TIME 缓冲区的下一个位置
    ; 加载冒号 ':'
    MOV    AL, ':'
    MOV    [BX], AL
    INC    BX
    ; 加载秒钟值
    MOV    AL, DL              ; 将秒钟值加载到 AL 寄存器中
    CALL    BCDtoASC           ; 调用 BCDtoASC 子程序将秒钟值转换为 ASCII 码
    ADD    BX, 2               ; 将 BX 寄存器增加 2，指向 TIME 缓冲区的下一个位置

```

```

;字符 '$'
MOV  AX, '$'
MOV  [BX], AL
;压栈保护现场
PUSH  BX
PUSH  CX
PUSH  DX
;显示
;清屏
MOV  AH,06H          ; 调用功能号 06H
MOV  AL,0
MOV  CH,0
MOV  CL,0
MOV  DH,24
MOV  DL,79
MOV  BH,01111000B    ; 设置白底黑字不闪烁
INT  10H
;输出
MOV  AH,13H          ; 调用功能号 13H
LEA  BP,TIME
MOV  DH,05H
MOV  DL,08H
MOV  AL,01H
MOV  CX,10
MOV  BL,01111000B    ; 设置黑底白字不闪烁
MOV  BH,00H
INT  10H
;弹出恢复现场
POP  DX
POP  CX
POP  BX
RET                  ; 返回到调用该子程序的位置
SHOW  ENDP

;BCDtoASC 子程序定义
BCDtoASC  PROC
    MOV  CL, AL;      暂存 AL
    SHR  AL, 1
    SHR  AL, 1
    SHR  AL, 1
    SHR  AL, 1
    OR   AL, 30H;     十位展开成 ASC，恢复
    MOV  [BX], AL
    MOV  AL, CL

```



```

    AND    AL, 0FH
    OR     AL, 30H;    个位展开成 ASCII, 恢复
    MOV    [BX+1], AL
    RET
BCDtoASC ENDP

CODE     ENDS
END      START

```

### 3. 程序设计思路:

#### (1) 数据段定义:

定义了数据段 “DATA”，其中包括存储时间的缓冲区 “TIME”，错误提示信息 “ERROR1” 和 “ERROR2”，还定义了堆栈段 “STA”，用于存放堆栈数据。

#### (2) 程序入口 “START”:

- ① 将 “DATA” 段的地址加载到 “DS” 和 “ES” 寄存器中。
- ② 输出冒号到屏幕上。
- ③ 输入时间。
- ④ 调用子程序进行输入错误和时间错误的判断。
- ⑤ 进入循环，进行时钟显示和更新。

#### (3) 输入错误判断子程序 “INPUTError”:

调用了 “NUMError” 和 “COLONError” 子程序逐个判断输入的时间格式是否正确。若格式错误，输出错误信息并跳回程序开始处。

#### (4) 数字判断子程序 “NUMError” 和冒号判断子程序 “COLONError”:

判断输入的字符是否为数字或冒号，若不是则置错误标志。

#### (5) “ASCtoBCD” 子程序:

将输入的 ASCII 码转换为 BCD 码，以便进行时间的处理和显示。具体有:

- ① 从存储输入时间的数组中取出 ASCII 码。
- ② 调用辅助子程序 “TRAN” 对每个时间单位进行转换。
- ③ 将转换后的 BCD 码存储到相应的位置，分别存储时、分、秒。

(6) 辅助子程序 “TRAN”：

辅助 ASCtoBCD 子程序进行 ASC 码到 BCD 码的转换。具体有：

- ① 通过位操作将 ASCII 码转换为 BCD 码。
- ② 对于个位数字，直接进行位操作处理。
- ③ 对于十位数字，先左移四位，然后与个位数字相加，得到 BCD 码。
- ④ 最后将得到的 BCD 码存储到指定位置

(6) 时间错误判断子程序 “TIMEError”：

检查输入的时间是否合法，包括时、分、秒的范围；若不合法，输出错误信息并跳回程序开始处。

(7) 延迟函数 “MDELAY”：

实现简单的延迟功能，用于每秒钟的计时。具体有：

- ① 保存当前环境，包括 AX、BX、CX、DX 寄存器的值。
- ② 使用 “INT 21H, AH=2CH” 系统调用获取当前系统时间，其中 “DH” 寄存器存储秒数。
- ③ 将秒数加 1，并进行与 60 的比较，若等于 60 则置 0，表示进入下一分钟。
- ④ 每次增加秒数后，再次获取系统时间，与之前备份的秒数进行比较，直到相等，表示延迟完成。

- ⑤ 恢复之前保存的环境，包括 AX、BX、CX、DX 寄存器的值。

(8) 时间加一子程序 “TIMECHANGE”：

将时间加一秒，处理了进位情况。具体有：

- ① 获取当前的秒数，并加 1。
- ② 使用 “DAA” 指令进行压缩 BCD 码运算的重要流程，确保 BCD 码的正确性。
- ③ 检查秒数是否等于 60，若是，则将秒数置 0，并将分钟加 1。
- ④ 检查分钟是否等于 60，若是，则将分钟置 0，并将小时加 1。
- ⑤ 检查小时是否等于 24，若是，则将小时置 0，表示进入下一天。

(9) 显示时间子程序 “SHOW”：

将当前的 BCD 码时间转换为可读的 ASCII 码并在屏幕上显示。具体有：

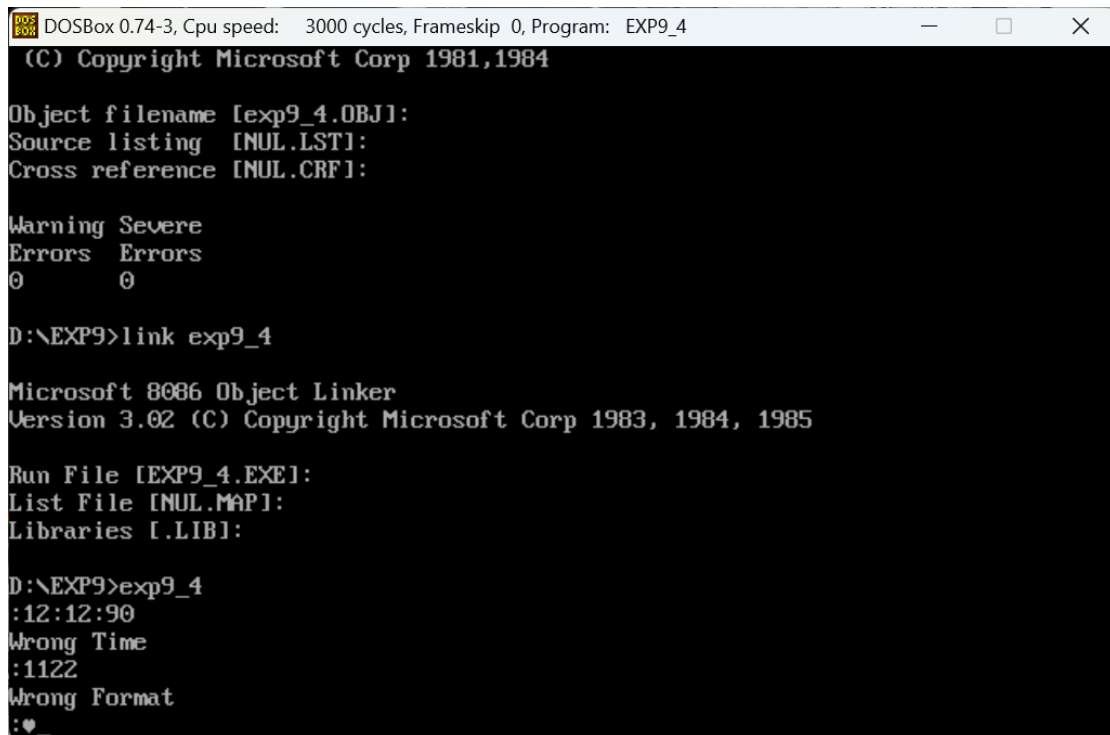
- ① 依次处理 BCD 码的时、分、秒，转换为对应的 ASCII 码，并存储到时间缓冲区中。
- ② 使用 “INT 10H” 的 BIOS 调用，清屏并输出时间信息到屏幕上。

(10) “BCDtoASC” 子程序：

将 BCD 码转换为 ASCII 码。具体有：

- ① 对于时钟的十位和个位，分别进行处理，将 BCD 码转换为对应的 ASCII 码。
- ② 使用位操作将 BCD 码转换为 ASCII 码。
- ③ 存储转换后的 ASCII 码到对应的位置，用于显示。

#### 4.实验结果：



```
DOSBox 0.74-3, Cpu speed: 3000 cycles, Frameskip 0, Program: EXP9_4
(C) Copyright Microsoft Corp 1981,1984
Object filename [exp9_4.OBJ]:
Source listing [NUL.LST]:
Cross reference [NUL.CRF]:

Warning Severe
Errors Errors
0 0

D:\EXP9>link exp9_4

Microsoft 8086 Object Linker
Version 3.02 (C) Copyright Microsoft Corp 1983, 1984, 1985

Run File [EXP9_4.EXE]:
List File [NUL.MAP]:
Libraries [.LIB]:

D:\EXP9>exp9_4
:12:12:90
Wrong Time
:1122
Wrong Format
:♥_
```

实现了检验输入是否出错，并相应报错的功能。



```
DOSBox 0.74-3, Cpu speed: 3000 cycles, Frameskip 0, Program: EXP9_4

:12:12:17
```

实现了输出清屏，设置字体和背景颜色，不换行并不停地计时的功能。

## 二、实验总结（实验中遇到的问题、解决方法和实验收获）

1. 很多功能重复的片段，可以单独写成子程序，简化代码，增强可读性，可理解成高级语言里的函数调用，比较便捷易懂。

2. 问题：如何将十进制码转换为 ASCII 码？

解决方法：十进制码的每一位和该数字所对应的 ASCII 码其实只差一个 30H，所以可通过“与”指令和移位指令将十进制所对应的 BCD 码的高四位与低四位分别提取出，并分别加上 30H 转换为 ASCII 码后再依次存入字符串空间即可。

3. 问题：使用条件转移指令时出现超出转移范围的情况

解决方法：更改各个代码段的次序，使条件转移指令和转移位置尽量靠近。同时，多用循环指令可以减少使用代码段空间。

## 三、回答思考题：

1. 时钟程序中存在时间误差吗？若有误差，其来源在何处？如何进行误差校正？

答：在按照实验原理进行设计的程序中存在实验误差，误差的来源主要是延时函数中用很多很多次减法来延迟时间。同时也要注意，在执行其他代码时也需要占时钟周期。可以通过多次调整循环次数，并一分钟计时计算误差，目前循环次数为  $0FFFFH \times 16H$ ，测得误差为 1.5 秒左右，较接近真实情况。误差校正需计算程序本身的运行时间，改变相应的计时次数，但误差无法被完全消除。