山东大学 计算机科学与技术 学院

汇编语言 课程实验报告

学号：202200130053	姓名：陈红瑞	班级：3 班
实验题目：实验 5：例 3.1，3.3，3.9		
实验学时：2	实验日期：20241111	
实验目的： 全面掌握汇编语言中的过程及其使用，进一步实践编程方法论。掌握通过全局变量、栈和寄存器传递过程参数与返回值的方法。掌握过程的模块化设计，及其嵌套与测试方法。掌握字节、字与双字数据的基本计算方法，以及部分系统调用。		
实验环境：Windows11、DOSBox-0.74、Masm64		
源程序清单： 1. 3_1.asm （示例 3.1） 2. 3_3.asm（示例 3.3） 3. 3_9.asm（示例 3.9）		
编译及运行结果： 1.例 3.1 下面是对程序的调试。 这里先设置了 CX 的值为 50，然后调用 shoot 函数。 进入到 shoot 函数，这里先在 DX 中设置了等待的时间，然后从 61H 端口中读取一个值，这里可以看出是 30H，将这个值取后 6 位，表示关断定时器通道 2 的门控，然后使用 XOR 指令来除法 61H 端口的第一位。		

后面再通过一些运算，得到一个脉冲宽度的计数值，这里为 7BH，然后通过一个循环等待一段时间，并重复这个过程，这里用 BX 存储重复的次数，结束后，再次关断定时器通道门控，并恢复端口，函数返回。

```
-u
076A:0100 B93200        MOV     CX,0032
076A:0103 51            PUSH    CX
076A:0104 E80E00        CALL    0115
076A:0107 B90040        MOV     CX,4000
076A:010A E2FE          LOOP    010A
076A:010C 59            POP     CX
076A:010D E2F4          LOOP    0103
076A:010F B048          MOV     AL,48
076A:0111 E661          OUT     61,AL
076A:0113 CD20          INT     20
076A:0115 BA4001        MOV     DX,0140
076A:0118 BB2000        MOV     BX,0020
076A:011B E461          IN      AL,61
076A:011D 24FC          AND     AL,FC
076A:011F 3402          XOR     AL,02
```

```
AX=FFFF  BX=0020  CX=0032  DX=0140  SP=FFFC  BP=0000  SI=0000  DI=0000
DS=075A  ES=075A  SS=0769  CS=076A  IP=011B   NV UP EI PL NZ NA PO NC
076A:011B E461          IN      AL,61
-t

AX=FF30  BX=0020  CX=0032  DX=0140  SP=FFFC  BP=0000  SI=0000  DI=0000
DS=075A  ES=075A  SS=0769  CS=076A  IP=011D   NV UP EI PL NZ NA PO NC
076A:011D 24FC          AND     AL,FC
```

```
AX=FF32  BX=0020  CX=0003  DX=9388  SP=FFFC  BP=0000  SI=0000  DI=0000
DS=075A  ES=075A  SS=0769  CS=076A  IP=0129   NV UP EI NG NZ NA PE NC
076A:0129 D3CA          ROR     DX,CL
-t

AX=FF32  BX=0020  CX=0003  DX=1271  SP=FFFC  BP=0000  SI=0000  DI=0000
DS=075A  ES=075A  SS=0769  CS=076A  IP=012B   NV UP EI NG NZ NA PE NC
076A:012B 8BCA          MOV     CX,DX
-t

AX=FF32  BX=0020  CX=1271  DX=1271  SP=FFFC  BP=0000  SI=0000  DI=0000
DS=075A  ES=075A  SS=0769  CS=076A  IP=012D   NV UP EI NG NZ NA PE NC
076A:012D 81E1FF01      AND     CX,01FF
-t

AX=FF32  BX=0020  CX=0071  DX=1271  SP=FFFC  BP=0000  SI=0000  DI=0000
DS=075A  ES=075A  SS=0769  CS=076A  IP=0131   NV UP EI PL NZ NA PE NC
076A:0131 83C90A        OR      CX,+0A
-t

AX=FF32  BX=0020  CX=007B  DX=1271  SP=FFFC  BP=0000  SI=0000  DI=0000
DS=075A  ES=075A  SS=0769  CS=076A  IP=0134   NV UP EI PL NZ NA PE NC
076A:0134 E2FE          LOOP    0134
```

对于上面执行的函数，这里还需要执行多次。在调用完函数后，先通过一个循环作为下一次发出声音的延迟，循环的次数为 4000H。然后再从栈中重新获得枪声此数的值，并执行发出枪声这一部分的循环。如果直接将剩下的循环全部执行完，会直接将一段连续的枪声输出。最后，恢复输出端口，并使用 INT 20H 来返回到 DOS。

```
-t

AX=FF30  BX=0000  CX=4000  DX=67A2  SP=FFFE  BP=0000  SI=0000  DI=0000
DS=075A  ES=075A  SS=0769  CS=076A  IP=010A   NV UP EI PL NZ NA PE NC
076A:010A E2FE          LOOP    010A
```

```
-g10f

AX=FF00  BX=0000  CX=0000  DX=67A2  SP=0000  BP=0000  SI=0000  DI=0000
DS=075A  ES=075A  SS=0769  CS=076A  IP=010F   NV UP EI PL ZR NA PE NC
076A:010F B048          MOV     AL,48
```

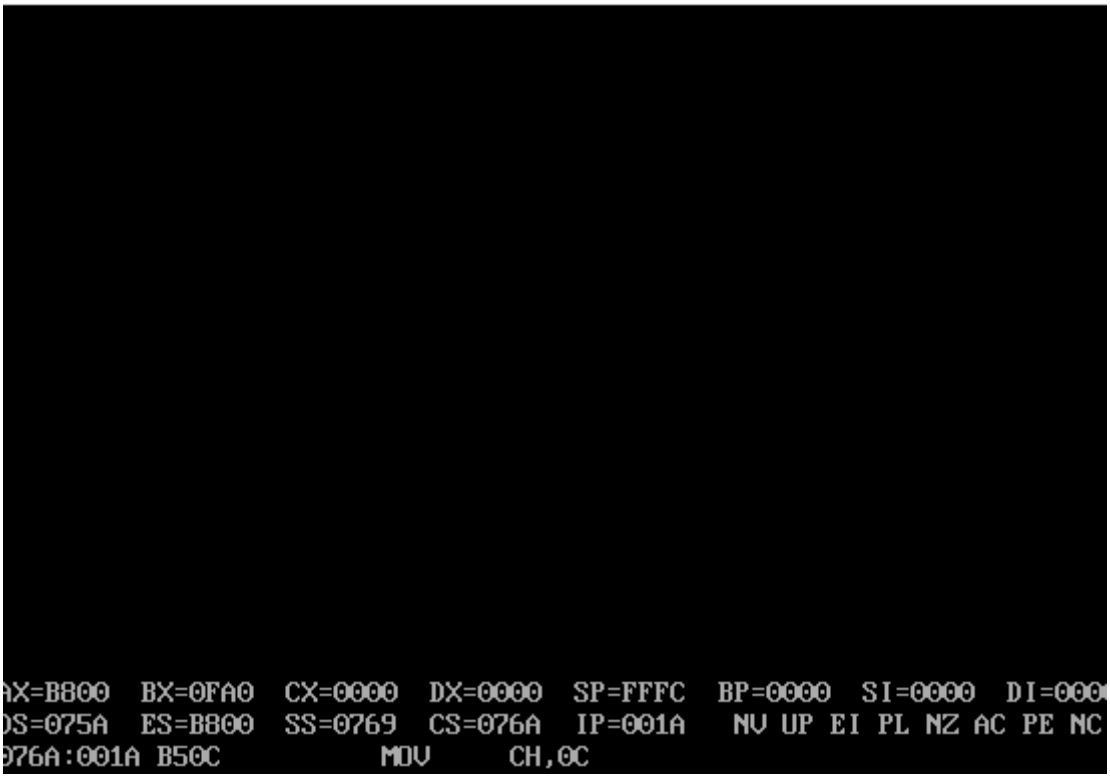
2. 例 3.3

下面是对程序的调试。

这里先定义了数据段和附加段的地址，然后在 CX 中设置计数值，即要处理的像素点总数，然后通过一个循环将附加段中的内容进行依次赋值，其中奇数地址为属性，这里设置为 07，偶数地址设置显示的字符，这里为 00。

```
C:\>debug 3_3.exe
-u
076A:0000 1E            PUSH    DS
076A:0001 2BC0          SUB     AX,AX
076A:0003 50            PUSH    AX
076A:0004 B800B8        MOV     AX,B800
076A:0007 8EC0          MOV     ES,AX
076A:0009 B9D007        MOV     CX,07D0
076A:000C BB0000        MOV     BX,0000
076A:000F 26            ES:
076A:0010 C78700000007  MOV     WORD PTR [BX+0000],0700
076A:0016 43            INC     BX
076A:0017 43            INC     BX
076A:0018 E2F5          LOOP    000F
076A:001A B50C          MOV     CH,0C
076A:001C B128          MOV     CL,28
076A:001E B400          MOV     AH,00
```

```
-d es:0
B800:0000  30 07 37 07 35 07 41 07-3A 07 42 07 38 07 32 07   0.7.5.A.:.B.8.2
B800:0010  30 07 20 07 20 07 30 07-30 07 20 07 30 07 30 07   0. . .0.0. .0.0
B800:0020  20 07 30 07 30 07 20 07-30 07 30 07 20 07 30 07    .0.0. .0.0. .0
B800:0030  30 07 20 07 30 07 30 07-20 07 30 07 30 07 20 07   0. .0.0. .0.0.
B800:0040  30 07 30 07 2D 07 30 07-30 07 20 07 30 07 30 07   0.0.-.0.0. .0.0
B800:0050  20 07 30 07 30 07 20 07-30 07 30 07 20 07 30 07    .0.0. .0.0. .0
B800:0060  20 07 20 07 20 07 20 07-20 07 20 07 20 07 20 07    . . . . . . . .
B800:0070  20 07 20 07 20 07 20 07-20 07 20 07 20 07 20 07    . . . . . . . .
```



然后设置函数和列数，这里在 CH 和 CL 中分别存储行数/2 和列数/2，用来表示初始时的位置为中心点。

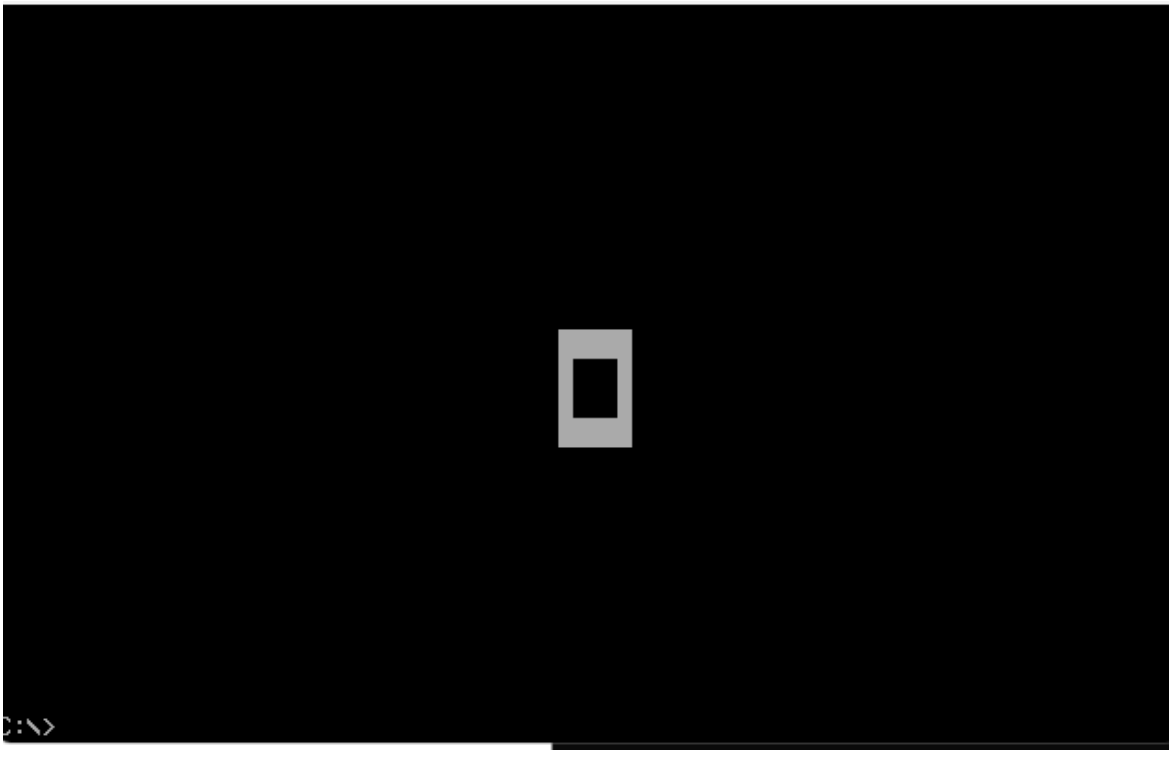
下面开始对屏幕上的光标进行处理，这里先调用 int 16h 中的功能 00H 来实现读取键盘的一个字符，然后会依次将这个字符与可能的结果进行比较，首先判断是否为 ESC 键，如果是就直接退出。后面再依次与不同的方向键进行比较，如果是某一个方向键，就按照方向来做出对应的操作，其中 CH 为行号，CL 为列号，根据对应的方向来将 CH 或 CL 进行加 1 或减 1，然后根据最新的 CX 中的值来计算出要修改的像素点在附加段中的地址，然后等待读取下一个字符并重复这个过程。

```
AX=B800  BX=0FA0  CX=0C28  DX=0000  SP=FFFC  BP=0000  SI=0000  DI=0000
DS=075A  ES=B800  SS=0769  CS=076A  IP=001E   NV UP EI PL NZ AC PE NC
076A:001E B400          MOV     AH,00
-t

AX=0000  BX=0FA0  CX=0C28  DX=0000  SP=FFFC  BP=0000  SI=0000  DI=0000
DS=075A  ES=B800  SS=0769  CS=076A  IP=0020   NV UP EI PL NZ AC PE NC
076A:0020 CD16          INT     16
```

下面是程序的运行结果。

这里刚开始光标在中间位置，此时如果按下除 ESC 以外的任意键均可以将中间点的位置变为白色，然后每次按下任意方向键，会将光标移动后的位置进行更改并将新的位置变为白色，如果光标所在的位置已经是白色，那么屏幕将保持不变。按下 ESC 后，会退出程序，返回 DOS。



3. 例 3.9

先对程序进行调试。

如图，这里先设置了 DS 和 ES 的值，然后再使用 INT 21H 中的功能

35H 来实现获取中断入口地址，这里将 AL 的值设置为 09H，表示中断来自键盘，中断号的入口地址为 ES：00，然后将 ES 和 BX 的值保存到内存中。

```
-g11

AX=077A  BX=0000  CX=0457  DX=0000  SP=00FC  BP=0000  SI=0000  DI=0000
DS=077A  ES=075A  SS=076A  CS=0792  IP=0011   NV UP EI PL ZR NA PE NC
0792:0011 8EC0          MOV     ES,AX
-t

AX=077A  BX=0000  CX=0457  DX=0000  SP=00FC  BP=0000  SI=0000  DI=0000
DS=077A  ES=077A  SS=076A  CS=0792  IP=0013   NV UP EI PL ZR NA PE NC
0792:0013 B435          MOV     AH,35
```

```
-u
0792:0013 B435          MOV     AH,35
0792:0015 B009          MOV     AL,09
0792:0017 CD21          INT     21
0792:0019 8C069400      MOV     [0094],ES
0792:001D 891E9600      MOV     [0096],BX
0792:0021 1E            PUSH    DS
0792:0022 BA9207        MOV     DX,0792
0792:0025 8EDA          MOV     DS,DX
0792:0027 BA8501        MOV     DX,0185
0792:002A B009          MOV     AL,09
0792:002C B425          MOV     AH,25
0792:002E CD21          INT     21
0792:0030 1F            POP     DS
0792:0031 B435          MOV     AH,35
```

下面设置中断向量。这里将 kbint 函数对应的段地址保存到 DS 中，然后通过调用 INT 21H 中的功能 25H 来设置中断向量表，中断号也是 09H。然后再调用 INT 21H 中的功能 35H 来实现计时器功能。

```
-u
0792:0021 1E            PUSH    DS
0792:0022 BA9207        MOV     DX,0792
0792:0025 8EDA          MOV     DS,DX
0792:0027 BA8501        MOV     DX,0185
0792:002A B009          MOV     AL,09
0792:002C B425          MOV     AH,25
0792:002E CD21          INT     21
0792:0030 1F            POP     DS
0792:0031 B435          MOV     AH,35
0792:0033 B01C          MOV     AL,1C
0792:0035 CD21          INT     21
0792:0037 891E6E01      MOV     [016E],BX
0792:003B 8C067001      MOV     [0170],ES
0792:003F 1E            PUSH    DS
0792:0040 BA9207        MOV     DX,0792
```

然后使用 in 和 out 指令来实现对计时器和键盘的初始化。然后使用 INT 10H 中的功能 0 来设置显示器模式。然后在显示器上输出数据段中的语句，如图，这里会先输出* PLEASE PRACTISE TYPING *，然后输出第一个要进行打字的句子，这时会将内存中对应的时间，包括 hours,min,sec 都初始化为 0,然后调用 kbget 函数来实现获取键盘输入。

```
-d1b
077A:0010                                       20 20 20 20 20
077A:0020  20 20 20 2A 20 50 4C 45-41 53 45 20 50 52 41 43     * PLEASE PRAC
077A:0030  54 49 53 45 20 54 59 50-49 4E 47 20 2A 0D 0A 24   TISE TYPING *..$
077A:0040  00 00 31 32 33 34 35 36-37 38 39 30 2D 3D 08 00   ..1234567890-=..
077A:0050  71 77 65 72 74 79 75 69-6F 70 5B 5D 0D 00 61 73   qwertyuiop[]..as
077A:0060  64 66 67 68 6A 6B 6C 3A-00 00 00 00 7A 78 63 76   dfghjkl:....zxcv
077A:0070  62 6E 6D 2C 2E 2F 00 00-00 20 00 00 00 00 00 00   bnm,./... ......
077A:0080  00 00 00 00 00 00 00 37-38 39 2D 34 35 36 2B 31   .......789-456+1
077A:0090  32 33 30 2E 00 F0 87 E9-61 62 63                  230.....abc
```

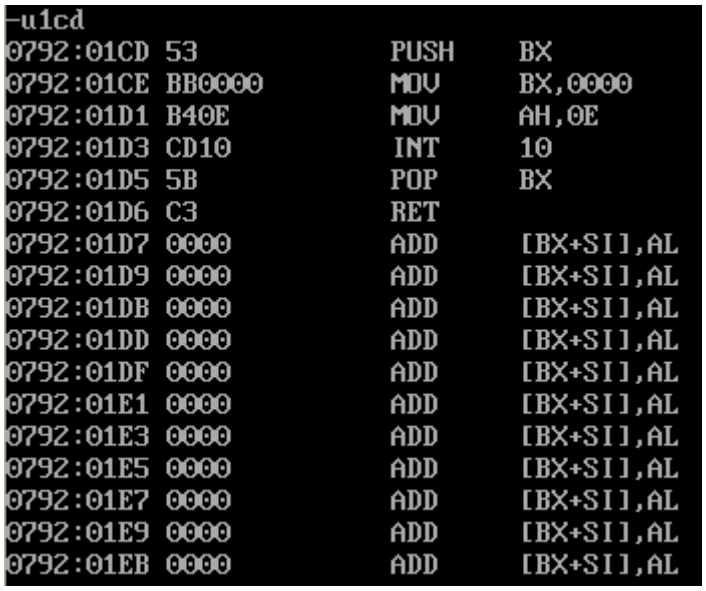
```
-d15c
077A:0150                                           98 00 BC 00                ....
077A:0160  E3 00 08 01 31 01 00 00-00 00 00 00 00 00 00 00   ....1...........
077A:0170  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ................
077A:0180  B8 6A 07 8E D0 BC 00 01-1E 2B C0 50 B8 7A 07 8E   .j.......+.P.z..
077A:0190  D8 8E C0 B4 35 B0 09 CD-21 8C 06 94 00 89 1E 96   ....5...!.......
077A:01A0  00 1E BA 92 07 8E DA BA-85 01 B0 09 B4 25 CD 21   .............%.!
077A:01B0  1F B4 35 B0 1C CD 21 89-1E 6E 01 8C 06 70 01 1E   ..5...!..n...p..
077A:01C0  BA 92 07 8E DA BA D8 00-B0 1C B4 25 CD 21 1F E4   ...........%.!..
077A:01D0  21 24 FC E6 21 B4 00 B0-03 CD 10 BA               !$..!.......
-d98
077A:0090                                 61 62 63 64 20 65 66 67          abcd efg
077A:00A0  68 20 69 6A 6B 6C 20 6D-6E 6F 70 20 71 72 73 74   h ijkl mnop qrst
077A:00B0  20 75 76 77 78 20 79 7A-2E 0D 0A 24 63 68 72 69    uvwx yz...$chri
077A:00C0  73 74 6D 61 73 20 69 73-20 61 20 74 69 6D 65 20   stmas is a time
077A:00D0  6F 66 20 6A 6F 79 20 61-6E 64 20 6C 6F 76 65 2E   of joy and love.
077A:00E0  0D 0A 24 73 74 6F 72 65-20 77 69 6E 64 6F 77 73   ..$store windows
077A:00F0  20 68 6F 6C 64 20 74 6F-67 73 20 61 6E 64 20 67    hold togs and g
077A:0100  69 66 74 73 2E 0D 0A 24-70 65 6F 70 6C 65 20 73   ifts...$people s
077A:0110  65 6E 64 20 63 68 72 69                           end chri
```

在 kbget 函数中，这里先关闭中断，然后通过指针来读取字符，如果当前指针为空，那就直接返回，否则就保存这个字符。

```
-u15b
0792:015B 53            PUSH    BX
0792:015C FA            CLI
0792:015D 8B1E1600      MOV     BX,[0016]
0792:0161 3B1E1800      CMP     BX,[0018]
0792:0165 750B          JNZ     0172
0792:0167 803E1A0000    CMP     BYTE PTR [001A],00
0792:016C 7511          JNZ     017F
0792:016E FB            STI
0792:016F 5B            POP     BX
0792:0170 EBE9          JMP     015B
0792:0172 8A870000      MOV     AL,[BX+0000]
0792:0176 43            INC     BX
0792:0177 83FB10        CMP     BX,+10
0792:017A 7203          JB      017F
```

在主函数中，这里还需要将刚才输入的字符显示出来，这里调用了 dispchar 函数，使用了 INT 10H 中的功能 0eH，实现将 AL 中的一个字符输出。然后判断这个字符是否为 0Dh，即输入的回车，如果是，就结束当前句子的输入，调用 disptime 函数来显示刚才使用的时间。

```
-u
0792:0085 FB            STI
0792:0086 E8D200        CALL    015B
0792:0089 F6061A0080    TEST    BYTE PTR [001A],80
0792:008E 7523          JNZ     00B3
0792:0090 50            PUSH    AX
0792:0091 E83901        CALL    01CD
0792:0094 58            POP     AX
0792:0095 3C0D          CMP     AL,0D
0792:0097 75ED          JNZ     0086
0792:0099 B00A          MOV     AL,0A
0792:009B E82F01        CALL    01CD
0792:009E E86C00        CALL    010D
0792:00A1 8D165601      LEA     DX,[0156]
```



```
-u1cd
0792:01CD 53            PUSH    BX
0792:01CE BB0000        MOV     BX,0000
0792:01D1 B40E          MOV     AH,0E
0792:01D3 CD10          INT     10
0792:01D5 5B            POP     BX
0792:01D6 C3            RET
0792:01D7 0000          ADD     [BX+SI],AL
0792:01D9 0000          ADD     [BX+SI],AL
0792:01DB 0000          ADD     [BX+SI],AL
0792:01DD 0000          ADD     [BX+SI],AL
0792:01DF 0000          ADD     [BX+SI],AL
0792:01E1 0000          ADD     [BX+SI],AL
0792:01E3 0000          ADD     [BX+SI],AL
0792:01E5 0000          ADD     [BX+SI],AL
0792:01E7 0000          ADD     [BX+SI],AL
0792:01E9 0000          ADD     [BX+SI],AL
0792:01EB 0000          ADD     [BX+SI],AL
```

在 disptime 函数，这里会依次输出分钟，秒和毫秒，这里先处理分钟数，从内存中取出分钟数，然后调用 bindec 函数来实现键二进制的数字转换为十进制的字符并显示，这里使用了 bindiv 函数，使用除法依次将百位,十位和个位的数字取出,并转换为 ASCII 字符,然后调用 INT 10H 中的功能 0EH 来将十进制字符输出。对秒和毫秒数的处理同理，中间用字符 ’ :’ 分开。其中对于毫秒数，需要先从内存中得到 count 的值，并将这个数乘 55 得到毫秒数在转换为十进制。

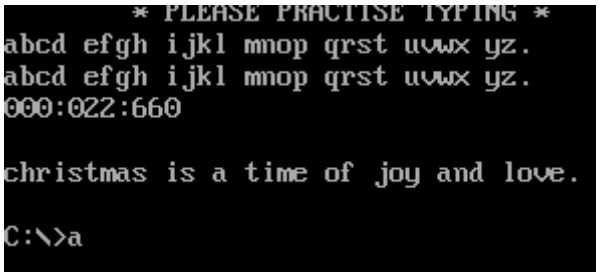
```
-u10d
0792:010D A16A01        MOV     AX,[016A]
0792:0110 E82400        CALL    0137
0792:0113 BB0000        MOV     BX,0000
0792:0116 B03A          MOV     AL,3A
0792:0118 B40E          MOV     AH,0E
0792:011A CD10          INT     10
0792:011C A16801        MOV     AX,[0168]
0792:011F E81500        CALL    0137
0792:0122 BB0000        MOV     BX,0000
0792:0125 B03A          MOV     AL,3A
0792:0127 B40E          MOV     AH,0E
0792:0129 CD10          INT     10
0792:012B 8B1E6601      MOV     BX,[0166]
```

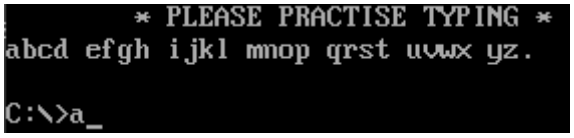
```
-u137
0792:0137 B96400        MOV     CX,0064
0792:013A E80D00        CALL    014A
0792:013D B90A00        MOV     CX,000A
0792:0140 E80700        CALL    014A
0792:0143 B90100        MOV     CX,0001
0792:0146 E80100        CALL    014A
0792:0149 C3            RET
0792:014A BA0000        MOV     DX,0000
0792:014D F7F1          DIV     CX
0792:014F BB0000        MOV     BX,0000
0792:0152 0430          ADD     AL,30
0792:0154 B40E          MOV     AH,0E
0792:0156 CD10          INT     10
```

执行完上面的过程后，会更新指向参考的字符串的指针，如果已经输出了 5 个句子，就重新初始化屏幕，从上面开始。如果中间有输入的字符为 ESC，就结束输出，重置中断向量。

对这个程序的运行结果如图。

这里先按照上面的句子输入字符，输入完成后按下 ENTER 键，这时会显示使用的时间，其中三个数字分别表示分钟，秒和毫秒，同时会进入到下一个句子的输入，当输入完第五个句子后，按下 ENTER 键会从第一行开始，重新显示第一个句子。





问题及收获：

1. 通过这次实验，学会实现发出声音的方法。
2. 这次实验掌握了通过方向键控制屏幕的方法，这里使用方向键来显示轨迹，使用到的 DOS 调用为 INT 16H，从键盘中读取一个字符，并根据方向键来判断是哪一个方向并对寄存器的值进行改变。
3. 掌握了对键盘和计时器的中断向量的实现。这里需要通过开中断来实现键盘输入，然后通过关中断来对输入的单个字符进行处理，包括保存指针以及显示，并判断是否标志结束。这里通过 INT 21H 中的功能 35H 实现自编中断向量，实现这个过程需要先保存当前的段地址，然后设置中断例行程序的段地址，并通过功能 25H 来设置中断向量功能调用。这里实现的键盘和计时器的中断。