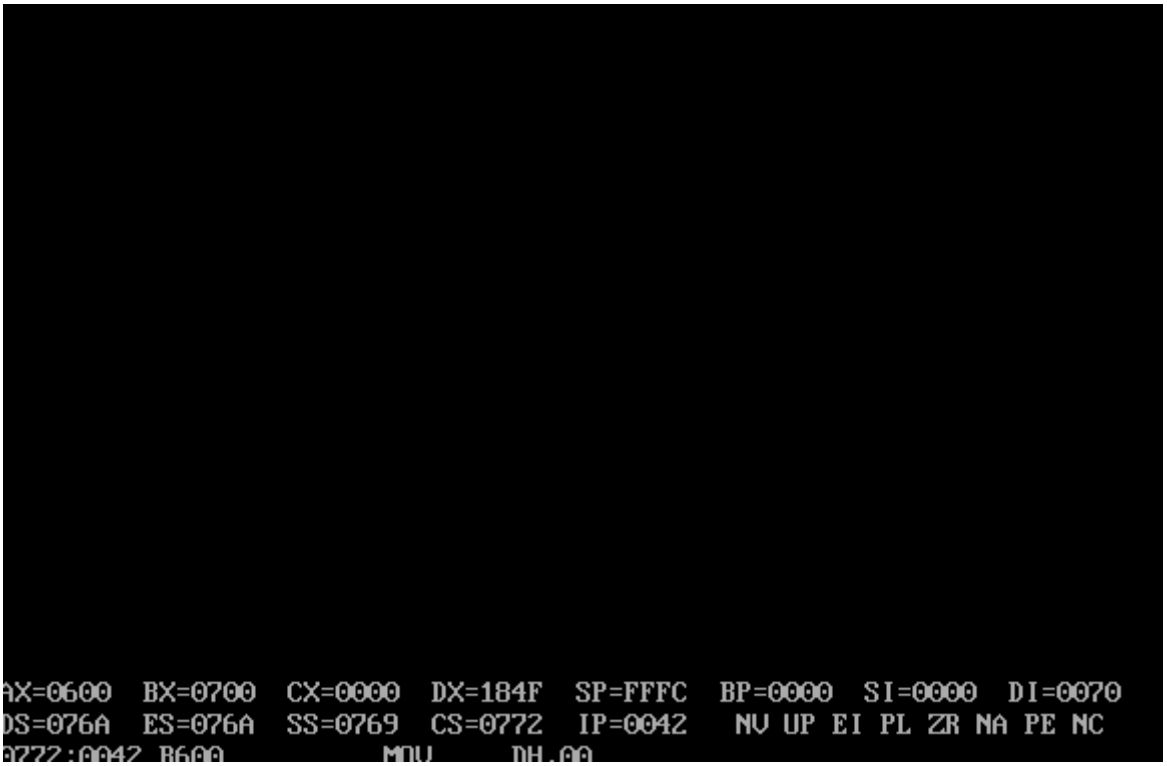
# 山东大学 计算机科学与技术 学院

# 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 6：例 3.8，例 4.1 | | |
| 实验学时：2 | 实验日期：20241118 | |
| 实验目的： 以键盘读取文件操作为例，掌握进行 BIOS 与 DOS 调用的思路。掌握 DOS 下文件的读取、写入与显示，以及使用 BIOS 读取键盘的方法。掌握输入缓冲区的基本设计与使用方法；熟悉常规命令行程序的人机交互逻辑编写。 | | |
| 实验环境：Windows11、DOSBox-0.74、Masm64 | | |
| 源程序清单：<br>1. 3_8.asm （示例 3.8）<br>2. 4_1.asm（示例 4.1） | | |
| 编译及运行结果：<br>1.例 3.8<br>下面是对程序的调试。<br>这里先将指针初始化为 0，包括缓冲区首部指针，末尾指针，行指针，列指针，并将当前最大行数初始化为 0，然后将 cntl 中的内容也初始化为 0，再将屏幕进行初始化， | | |

```
-u
0772:0000 1E            PUSH    DS
0772:0001 2BC0          SUB     AX,AX
0772:0003 50            PUSH    AX
0772:0004 B86A07        MOV     AX,076A
0772:0007 8ED8          MOV     DS,AX
0772:0009 8EC0          MOV     ES,AX
0772:000B C70672000000  MOV     WORD PTR [0072],0000
0772:0011 C606740000    MOV     BYTE PTR [0074],00
0772:0016 C606750000    MOV     BYTE PTR [0075],00
0772:001B C70670000000  MOV     WORD PTR [0070],0000
```




下面调用宏来设置光标的位置，这里先设置 DH 和 DL 的值，用于表示光标的行号和列号，然后用 BH 设置页号为 0，最后调用 INT 10H 的功能 02H 来实现初始化光标位置为左上角。然后使用 INT 16H 中的功能 00H 来实现通过键盘输入一个字符并读取，

```
-u
0772:0042 B600          MOV     DH,00
0772:0044 B200          MOV     DL,00
0772:0046 B700          MOV     BH,00
0772:0048 B402          MOV     AH,02
0772:004A CD10          INT     10
0772:004C B400          MOV     AH,00
0772:004E CD16          INT     16
0772:0050 3C1B          CMP     AL,1B
0772:0052 7501          JNZ     0055
0772:0054 CB            RETF
0772:0055 80FC4B        CMP     AH,4B
0772:0058 7408          JZ      0062
0772:005A 80FC4D        CMP     AH,4D
0772:005D 7406          JZ      0065
0772:005F EB07          JMP     0068
0772:0061 90            NOP
```

输入完一个字符后，会判断是否为左右方向键，如果是，就跳转到对应的位置，否则，就执行下面的插入字符的操作。

```
AX=1F73  BX=0000  CX=0000  DX=0000  SP=FFFC  BP=0000  SI=0000  DI=0070
DS=076A  ES=076A  SS=0769  CS=0772  IP=0055   NV UP EI PL NZ AC PO NC
0772:0055 80FC4B        CMP     AH,4B
```

这里会先判断缓冲区首尾指针是否在同一个位置，如果是，就直接将当前的字符插入到缓冲区，这里先输入了一个字符 s ，将其存入内存中，然后将对应的行的字符计数加 1，并将列指针加 1。下面再调用一个过程，实现将刚才输入的内容进行输出。

```
-t

AX=1F73  BX=0000  CX=0000  DX=0000  SP=FFFC  BP=0000  SI=0000  DI=0070
DS=076A  ES=076A  SS=0769  CS=0772  IP=0082   NV UP EI PL ZR NA PE NC
0772:0082 88870000      MOV     [BX+0000],AL                    DS:0000=20
-t

AX=1F73  BX=0000  CX=0000  DX=0000  SP=FFFC  BP=0000  SI=0000  DI=0070
DS=076A  ES=076A  SS=0769  CS=0772  IP=0086   NV UP EI PL ZR NA PE NC
0772:0086 FF067000      INC     WORD PTR [0070]                 DS:0070=0000
-d0
076A:0000  73 20 20 20 20 20 20 20-20 20 20 20 20 20 20 20   s
076A:0010  20 20 20 20 20 20 20 20-20 20 20 20 20 20 20 20
076A:0020  20 20 20 20 20 20 20 20-20 20 20 20 20 20 20 20
076A:0030  20 20 20 20 20 20 20 20-20 20 20 20 20 20 20 20
076A:0040  20 20 20 20 20 20 20 20-20 20 20 20 20 20 20 20
076A:0050  20 20 20 20 20 20 20 20-20 20 20 20 20 20 20 20
076A:0060  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ................
076A:0070  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ................
```

```
-t

AX=1F73  BX=0000  CX=0000  DX=0000  SP=FFFC  BP=0000  SI=0000  DI=0070
DS=076A  ES=076A  SS=0769  CS=0772  IP=00E5   NV UP EI PL NZ AC PE NC
0772:00E5 8A1E7500      MOV     BL,[0075]                          DS:0075=00
-u
0772:00E5 8A1E7500      MOV     BL,[0075]
0772:00E9 32FF          XOR     BH,BH
0772:00EB FE876000      INC     BYTE PTR [BX+0060]
0772:00EF FE067400      INC     BYTE PTR [0074]
0772:00F3 E87D00        CALL    0173
0772:00F6 8A367500      MOV     DH,[0075]
0772:00FA 8A167400      MOV     DL,[0074]
0772:00FE B700          MOV     BH,00
0772:0100 B402          MOV     AH,02
0772:0102 CD10          INT     10
0772:0104 E945FF        JMP     004C
```

进入到 dispbf 的过程内，这里同样实现了使用宏来设置光标，将光标设置在了左上角，然后将缓冲区的所有的字符依次输出，如果遇到了回车，就将 CRLF 一起输出，这里调用的 INT 10H 的功能 0EH，可以实现将一个字符显示在屏幕上，并将光标后移。输出完成后，函数返回。

```
-u
0772:0173 BB0000        MOV     BX,0000
0772:0176 B96000        MOV     CX,0060
0772:0179 B600          MOV     DH,00
0772:017B B200          MOV     DL,00
0772:017D B700          MOV     BH,00
0772:017F B402          MOV     AH,02
0772:0181 CD10          INT     10
0772:0183 8A870000      MOV     AL,[BX+0000]
0772:0187 53            PUSH    BX
0772:0188 BBBC02        MOV     BX,02BC
0772:018B B40E          MOV     AH,0E
0772:018D CD10          INT     10
0772:018F 5B            POP     BX
0772:0190 3C0D          CMP     AL,0D
0772:0192 7506          JNZ     019A
```

下面再考虑输入了方向键的情况。这里输入了左方向键，因此不会插入对应的字符，但是需要重新设置光标的位置，这里先判断光标原本是否在第一列，如果不是，就直接将列对应的指针减 1，否则就继续判断是否在第一行，如果是，这一次的移动光标是无效的，直接读取下一个字符的输入，否则将行指针减 1，并将列指针指向上一行的最后一列。

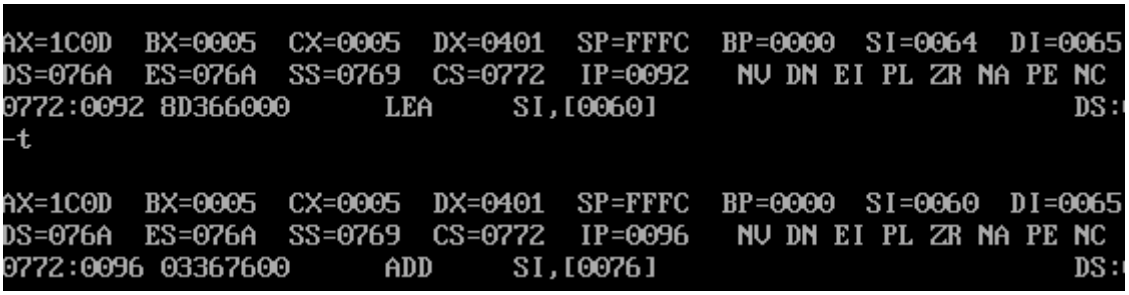
右方向键的处理同理。

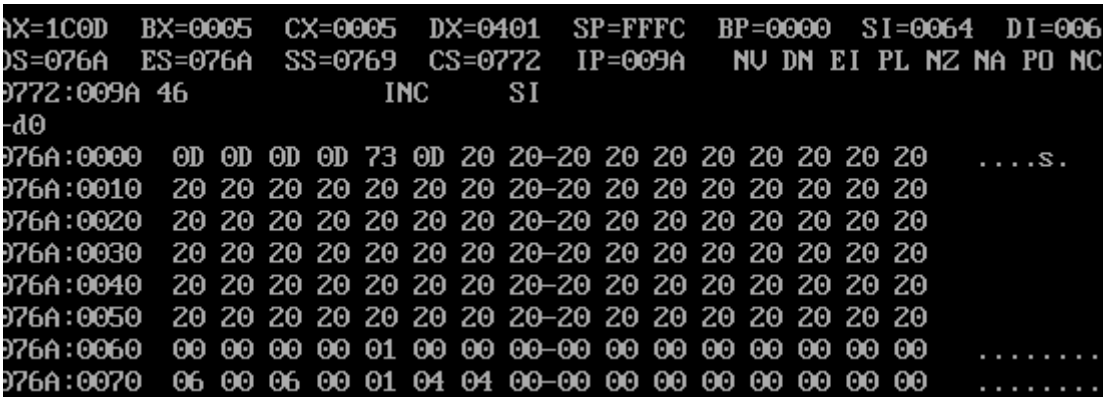
```
-u
0772:0055 80FC4B        CMP     AH,4B
0772:0058 7408          JZ      0062
0772:005A 80FC4D        CMP     AH,4D
0772:005D 7406          JZ      0065
0772:005F EB07          JMP     0068
0772:0061 90            NOP
0772:0062 E9A200        JMP     0107
0772:0065 E9D800        JMP     0140
0772:0068 8B1E7000      MOV     BX,[0070]
0772:006C 8B0E7200      MOV     CX,[0072]
0772:0070 3BD9          CMP     BX,CX
0772:0072 740E          JZ      0082
0772:0074 8D3E0000      LEA     DI,[0000]
-t

AX=4B00  BX=0060  CX=0000  DX=0000  SP=FFFC  BP=0000  SI=0000  DI=0070
DS=076A  ES=076A  SS=0769  CS=0772  IP=0058   NV UP EI PL ZR NA PE NC
0772:0058 7408          JZ      0062
-t

AX=4B00  BX=0060  CX=0000  DX=0000  SP=FFFC  BP=0000  SI=0000  DI=0070
DS=076A  ES=076A  SS=0769  CS=0772  IP=0062   NV UP EI PL ZR NA PE NC
0772:0062 E9A200        JMP     0107
```

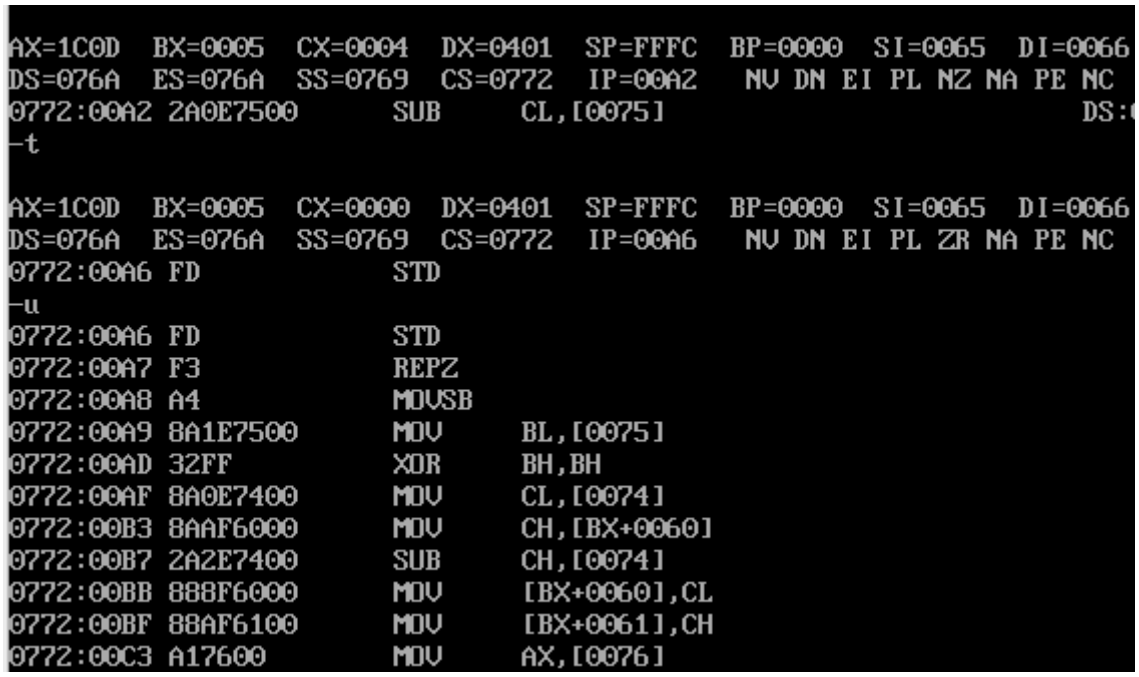
```
AX=4B00  BX=0060  CX=0000  DX=0000  SP=FFFC  BP=0000  SI=0000  DI=0070
DS=076A  ES=076A  SS=0769  CS=0772  IP=0107   NV UP EI PL ZR NA PE NC
0772:0107 803E740000    CMP     BYTE PTR [0074],00                DS:
-t

AX=4B00  BX=0060  CX=0000  DX=0000  SP=FFFC  BP=0000  SI=0000  DI=0070
DS=076A  ES=076A  SS=0769  CS=0772  IP=010C   NV UP EI PL ZR NA PE NC
0772:010C 7519          JNZ     0127
-t

AX=4B00  BX=0060  CX=0000  DX=0000  SP=FFFC  BP=0000  SI=0000  DI=0070
DS=076A  ES=076A  SS=0769  CS=0772  IP=010E   NV UP EI PL ZR NA PE NC
0772:010E 803E750000    CMP     BYTE PTR [0075],00                DS:
-t

AX=4B00  BX=0060  CX=0000  DX=0000  SP=FFFC  BP=0000  SI=0000  DI=0070
DS=076A  ES=076A  SS=0769  CS=0772  IP=0113   NV UP EI PL ZR NA PE NC
0772:0113 7428          JZ      013D
-t

AX=4B00  BX=0060  CX=0000  DX=0000  SP=FFFC  BP=0000  SI=0000  DI=0070
DS=076A  ES=076A  SS=0769  CS=0772  IP=013D   NV UP EI PL ZR NA PE NC
0772:013D E90CFF        JMP     004C
```

下面考虑出现了回车的情况。由于输入了回车，因此在插入这个字符到缓冲区后，需要考虑将回车符所在的位置后面的的所有的行的行号加

1，这里先获得 cntl 的地址，并通过加法来获取其最后一个位置的地址。如图，此时的缓冲区一共有 6 个字符，此时需要将输入回车的后一个位置都向下移动一行，因此需要移动的行数为当前的最大行数减行指针，然后从最后一个位置开始依次向下移动，这里使用 STD 表示操作的地址从高到低，重复的次数保存在 CX 中。



此时还需要考虑输入的回车所在行的右边部分没有处理，因此需要根据当前行的 cntl 值以及当前的列指针来计算出需要移动到下一行的部分的长度，这里执行的操作为将当前行原来的总长度保存到 CH 中，然后将当前的列指针保存到 CL 中，然后将 CH 中的值与列指针的值相减，结果保存到 CH 中，这时 CH 中的是当前行修改后的 cntl 值，而 CL 中是下一行的 cntl 值。

下面还需要将屏幕上卷，这里使用 INT 16H 中的功能 06H，将左上角设置为回车符所在的行，然后将行指针和总行数加 1，最后执行和输入字符相同的操作。

```
AX=1C0D  BX=0004  CX=0000  DX=0401  SP=FFFC  BP=0000  SI=0065  DI=0066
DS=076A  ES=076A  SS=0769  CS=0772  IP=00AF   NV DN EI PL ZR NA PE NC
0772:00AF 8A0E7400      MOV     CL,[0074]                       DS:0
-t

AX=1C0D  BX=0004  CX=0001  DX=0401  SP=FFFC  BP=0000  SI=0065  DI=0066
DS=076A  ES=076A  SS=0769  CS=0772  IP=00B3   NV DN EI PL ZR NA PE NC
0772:00B3 8AAF6000      MOV     CH,[BX+0060]                    DS:0
-t

AX=1C0D  BX=0004  CX=0101  DX=0401  SP=FFFC  BP=0000  SI=0065  DI=0066
DS=076A  ES=076A  SS=0769  CS=0772  IP=00B7   NV DN EI PL ZR NA PE NC
0772:00B7 2A2E7400      SUB     CH,[0074]                       DS:0
-t

AX=1C0D  BX=0004  CX=0001  DX=0401  SP=FFFC  BP=0000  SI=0065  DI=0066
DS=076A  ES=076A  SS=0769  CS=0772  IP=00BB   NV DN EI PL ZR NA PE NC
0772:00BB 888F6000      MOV     [BX+0060],CL                    DS:0
```

下面运行代码。运行结果与 word 中的功能相同，按 ESC 后退出。

```
12
34
qwe
.,a
fs
C:\>
```

2. 例 4.1

对代码进行调试。

这里首先会将每一页显示的行数初始化为 12 行。缓冲区的大小为 200，因此将当前的指针指向 200 的位置表示缓冲区为空。

```
-u
0782:0000 1E            PUSH    DS
0782:0001 2BC0          SUB     AX,AX
0782:0003 50            PUSH    AX
0782:0004 B86A07        MOV     AX,076A
0782:0007 8ED8          MOV     DS,AX
0782:0009 C706000000C00 MOV     WORD PTR [0000],000C
0782:000F C706CC00C800  MOV     WORD PTR [00CC],00C8
0782:0015 E8BD00        CALL    00D5
0782:0018 E89F00        CALL    00BA
0782:001B 0BC0          OR      AX,AX
0782:001D 750A          JNZ     0029
0782:001F BA0A01        MOV     DX,010A
```

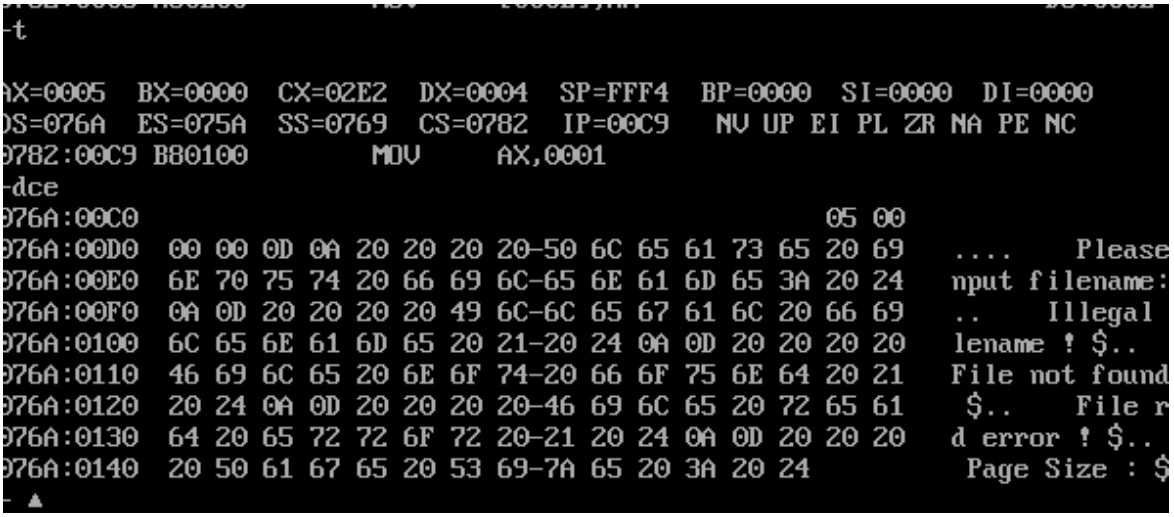
下面调用 getline 来获取访问的文件名。如图，这里先将内存中读取字符串并输出，然后使用 INT 21H 的功能 0AH 来输入文件名，并将其保存到内存中。

```
0782:00D7 51            PUSH    CX
0782:00D8 52            PUSH    DX
0782:00D9 BAD200        MOV     DX,00D2
0782:00DC B409          MOV     AH,09
0782:00DE CD21          INT     21
0782:00E0 BA0200        MOV     DX,0002
0782:00E3 B40A          MOV     AH,0A
0782:00E5 CD21          INT     21
0782:00E7 BA4E01        MOV     DX,014E
0782:00EA B409          MOV     AH,09
0782:00EC CD21          INT     21
0782:00EE 8A1E0300      MOV     BL,[0003]
0782:00F2 B700          MOV     BH,00
0782:00F4 C687040000    MOV     BYTE PTR [BX+0004],00
-dd2
076A:00D0       0D 0A 20 20 20 20-50 6C 65 61 73 65 20 69     ..    Please
076A:00E0  6E 70 75 74 20 66 69 6C-65 6E 61 6D 65 3A 20 24   nput filename:
076A:00F0  0A 0D 20 20 20 20 49 6C-6C 65 67 61 6C 20 66 69   ..    Illegal f
076A:0100  6C 65 6E 61 6D 65 20 21-20 24 0A 0D 20 20 20 20   lename ! $..
076A:0110  46 69 6C 65 20 6E 6F 74-20 66 6F 75 6E 64 20 21   File not found
076A:0120  20 24 0A 0D 20 20 20 20-46 69 6C 65 20 72 65 61    $..    File re
076A:0130  64 20 65 72 72 6F 72 20-21 20 24 0A 0D 20 20 20   d error ! $..
076A:0140  20 50 61 67 65 20 53 69-7A 65 20 3A 20 24 0A 0D    Page Size : $.
076A:0150  24 0A                                            $.
```

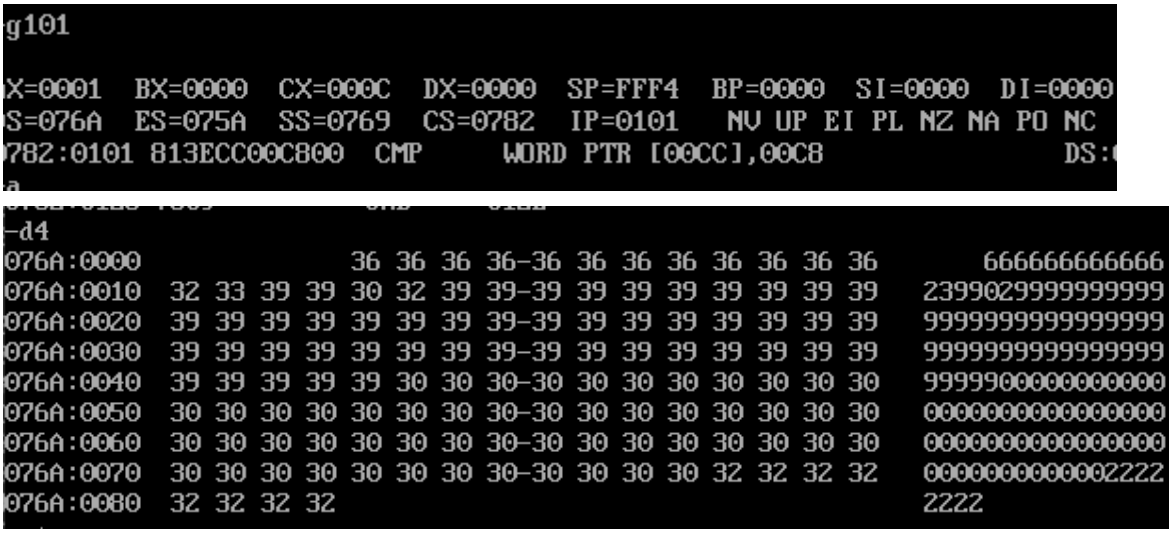
```
-ge7

   Please input filename: test.txt
AX=0A6A  BX=0000  CX=02E2  DX=0002  SP=FFF2  BP=0000  SI=0000  DI=0000
DS=076A  ES=075A  SS=0769  CS=0782  IP=00E7   NV UP EI PL ZR NA PE NC
0782:00E7 BA4E01        MOV     DX,014E
-d0
076A:0000  0C 00 50 08 74 65 73 74-2E 74 78 74 0D 00 00 00   ..P.test.txt..
076A:0010  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ................
076A:0020  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ................
076A:0030  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ................
076A:0040  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ................
076A:0050  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ................
076A:0060  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ................
076A:0070  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ................
-a
```

下面再调用 openf 过程，这里调用 INT 21H 中的功能 3DH，实现打开文件的操作，打开的方式为只读。然后将文件句柄保存到内存中，这里显示的文件句柄为 05H.

```
-u
0782:00BA 53            PUSH    BX
0782:00BB 51            PUSH    CX
0782:00BC 52            PUSH    DX
0782:00BD BA0400        MOV     DX,0004
0782:00C0 B000          MOV     AL,00
0782:00C2 B43D          MOV     AH,3D
0782:00C4 CD21          INT     21
0782:00C6 A3CE00        MOV     [00CE],AX
0782:00C9 B80100        MOV     AX,0001
0782:00CC 7303          JNB     00D1
0782:00CE B80000        MOV     AX,0000
0782:00D1 5A            POP     DX
0782:00D2 59            POP     CX
0782:00D3 5B            POP     BX
0782:00D4 C3            RET
```

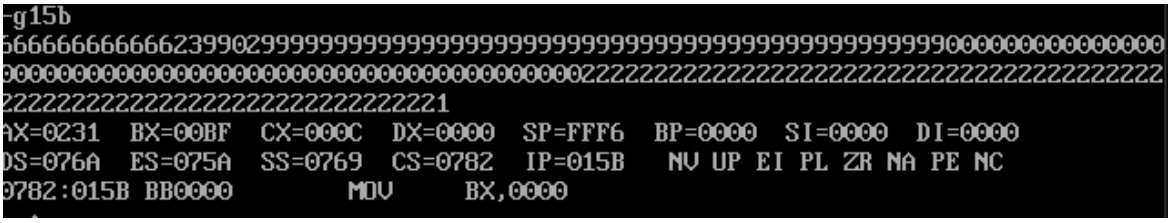



这里文件打开成功了，跳转到读取文件的部分，调用 read_block，这里会先判断缓冲区的指针是否为 200，即缓冲区是否为空，如果为空，就开始读取 200 个字符到缓冲区，这里需要调用 INT 21H 中的功能 3FH，这里根据 DX 中指定的数据段地址来作为缓冲区，然后从文件中读取 200 个字符，然后将当前缓冲区的指针设置为 0，表示缓冲区为满。




下面执行 show_block 过程，这里先将 cur 指针存入到 BX 中，然后，判断这个值是否为 200，如果是，说明缓冲区是空的，直接返回，否则读取一个字符，此时需要判断文件是否结束，由于文件结束符可能不会在缓冲区中出现，这里改成了缓冲区的指针的位置到达缓冲区最后一个

字符且不是 200 时，说明文件已经到了结束，因此需要结束对文件内容的输出，否则就显示这个字符，此外还需要判断字符是否为换行，如果是，就需要对当前页的剩余行数减 1，并结束这一轮的输出。这里的数据为一行且只有 191 个字符，因此会在这一行结束输出，次数 BX 中缓冲区的指针与缓冲区实际长度相等，文件到达末尾时，会将 BX 的值设置为 0。




返回后，判断 BX 中的值是否为 0，如果是 0 就返回到 DOS，否则就根据 CX 来判断这一页的输出是否结束，如果结束就调用 INT 21H 的 01H 功能来读取一个字符，并判断是否为空格或 p，如果是空格就在 CX 中重置剩余的页数，否则就调用 change_psize 过程。

当一页的内容输出完成后，输入的字符为 p，此时会进入到 change_psize 过程。这里会先使用 INT 21H 来实现将提示内容输出到屏幕中。然后获取一个输入的字符，这里输入的是 1。然后会判断这个字符是否为回车，如果输入的是数字，就将其减’0’来转换为二进制数字存储，然后保存到 CL 中。下面会再次获取一个输入，如果是回车，就将 CX 中的数字更新 psize 的值，否则就将原来 CX 中的数字乘 10 加上这次输入的数，得到一个两位十进制数，再保存到 CX 中，这里我输入的第二个数字为 3，会计算出 13 并保存到 CX 中。下面会重复这个过程，再次获取一次输入，这次输入回车，会将 CX 的值保存到 psize 中并返回。

```
-d13b
076A:0130                                                0A 0D 20 20 20
076A:0140  20 50 61 67 65 20 53 69-7A 65 20 3A 20 24 0A 0D    Page Size
076A:0150  24 0A 0D 2A 2A 2A 2A 2A-2A 2A 2A 2A 2A 2A 2A 2A   $..*********
076A:0160  2A 2A 2A 2A 2A 2A 2A 2A-2A 2A 2A 2A 2A 2A 2A 2A   **********
076A:0170  2A 0A 0D 24 00 00 00 00-00 00 00 00 00 00 00 00   *..$........
076A:0180  1E 2B C0 50 B8 6A 07 8E-D8 C7 06 00 00 0C 00 C7   .+.P.j......
076A:0190  06 CC 00 C8 00 E8 BD 00-E8 9F 00 0B C0 75 0A BA   ...........
076A:01A0  0A 01 B4 09 CD 21 EB 43-90 8B 0E 00 00 E8 CE 00   .....!.C....
076A:01B0  0B C0 75 0A BA 22 01 B4-09 CD 21                  ..u.."....!
```

```
-g7b
1
AX=0131  BX=0033  CX=0000  DX=013B  SP=FFF
DS=076A  ES=075A  SS=0769  CS=0782  IP=007
0782:007B 3C0D          CMP     AL,0D
-t

AX=0131  BX=0033  CX=0000  DX=013B  SP=FFF
DS=076A  ES=075A  SS=0769  CS=0782  IP=007
0782:007D 7430          JZ      00AF
-
```

```
AX=0101  BX=0033  CX=0000  DX=013B  SP=FFF2  BP=0000
DS=076A  ES=075A  SS=0769  CS=0782  IP=0081   NV UP E
0782:0081 8AC8          MOV     CL,AL
-t

AX=0101  BX=0033  CX=0001  DX=013B  SP=FFF2  BP=0000
DS=076A  ES=075A  SS=0769  CS=0782  IP=0083   NV UP E
0782:0083 B401          MOV     AH,01
-
```

```
0782:0091 8ACA          MOV     CL,DL
-t

AX=0101  BX=0033  CX=0003  DX=0103  SP=FFF2  BP=
DS=076A  ES=075A  SS=0769  CS=0782  IP=0093   NV
0782:0093 B30A          MOV     BL,0A
-t

AX=0101  BX=000A  CX=0003  DX=0103  SP=FFF2  BP=
DS=076A  ES=075A  SS=0769  CS=0782  IP=0095   NV
0782:0095 F6E3          MUL     BL
-t

AX=000A  BX=000A  CX=0003  DX=0103  SP=FFF2  BP=
DS=076A  ES=075A  SS=0769  CS=0782  IP=0097   NV
0782:0097 02C8          ADD     CL,AL
-t

AX=000A  BX=000A  CX=000D  DX=0103  SP=FFF2  BP=
DS=076A  ES=075A  SS=0769  CS=0782  IP=0099   NV
0782:0099 EBE8          JMP     0083
```

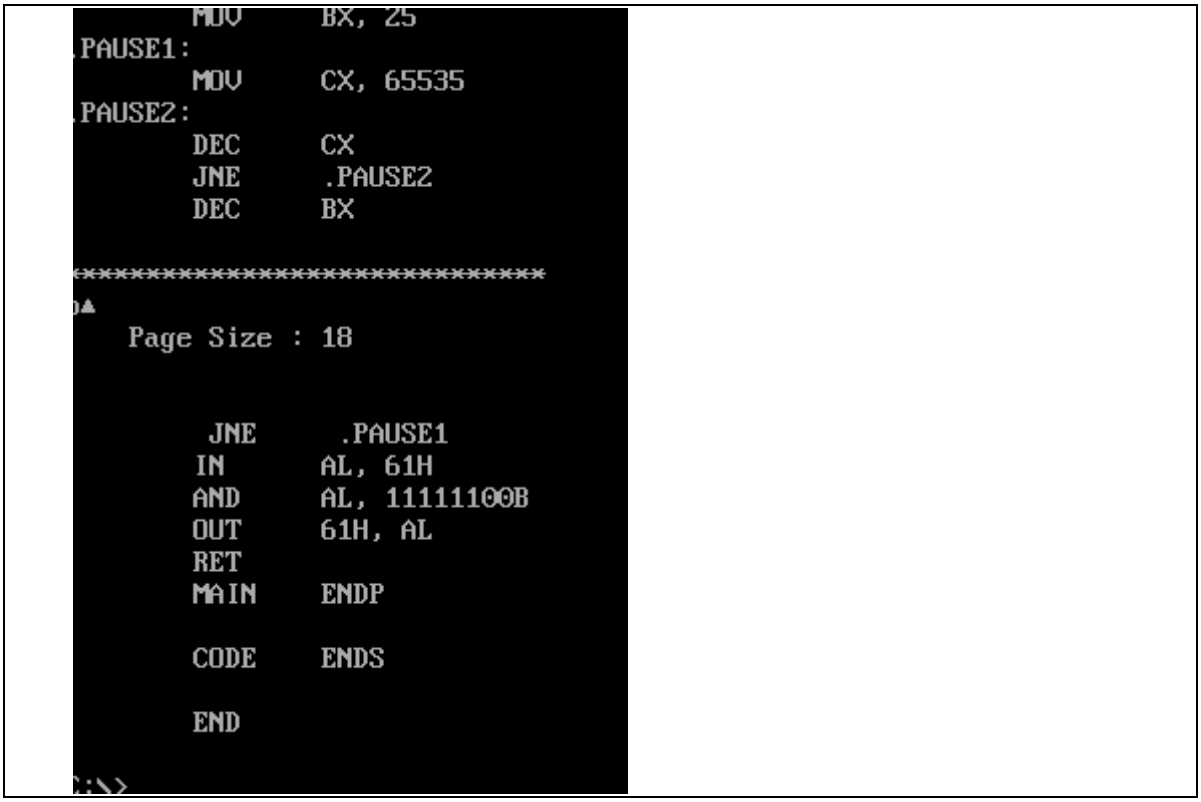
下面是对整个程序的运行。

如图，这里将 test.asm 文件作为访问的文件。

初始时每次显示 12 行内容，到达文件末尾后会直接返回。

```
C:\>4_1

    Please input filename: test.asm
        CODE    SEGMENT
        ASSUME  CS:CODE

        MAIN    PROC FAR
        PUSH    DS
        SUB     AX, AX
        PUSH    AX
START:
        MOV     AL, 182
        OUT     43H, AL
        MOV     AX, 4560
        OUT     42H, AL

********************************
```

```
        PUSH    DS
        SUB     AX, AX
        PUSH    AX
START:
        MOV     AL, 182
        OUT     43H, AL
        MOV     AX, 4560
        OUT     42H, AL

********************************
         MOV     AL, AH
        OUT     42H, AL
        IN      AL, 61H
        OR      AL, 00000011B
        OUT     61H, AL
        MOV     BX, 25
.PAUSE1:
        MOV     CX, 65535
.PAUSE2:
        DEC     CX
        JNE     .PAUSE2
        DEC     BX

********************************
```



```
        MOV     BX, 25
.PAUSE1:
        MOV     CX, 65535
.PAUSE2:
        DEC     CX
        JNE     .PAUSE2
        DEC     BX

**********************************
▲
    Page Size : 18


         JNE     .PAUSE1
        IN      AL, 61H
        AND     AL, 11111100B
        OUT     61H, AL
        RET
        MAIN    ENDP

        CODE    ENDS

        END

C:\>
```

问题及收获：

1. 通过第一个例程，掌握了宏的设计方法，这里通过宏来简化对光标的位置的设置。
2. 通过第二个例程，了解到实现文件读取时用到的 DOS 调用，这里使用的是 INT 21H 中的功能 3DH 和 3FH，分别用于打开文件和将文件中的指定长度的内容保存到内存中。
3. 这里修改了 4.1 中 BUG，了解了文件访问时的过程，这里使用了一种更简单的方法来判断文件是否到了末尾，即缓冲区出现的没有满但是指针已经到最后一个字符的情况时就可以说明文件已经访问完成。