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 7：例 2.8		
实验学时：2	实验日期：20241125	
实验目的： 掌握综合性汇编语言程序设计的方法。掌握递归算法的设计与汇编表示。全面回顾前述一切实验的内容。		
实验环境：Windows11、DOSBox-0.74、Masm64		
源程序清单： 2_8.asm		
编译及运行结果： 首先调试程序。 这里先调用过程 decibin，读取一个输入作为 N 的值，实验 INT 21H 中的功能 01H，通过键盘获取一个字符的输入，然后通过将读取到的 ASCII 字符减 30H 转换为一个数字，然后判断是否小于 0，以及是否大于 9 来判断是否为一个十进制数字，如果是的话，就将这个数存入到寄存器 AX 中，而原来的数在 BX 中，需要将两者交换后，用 mul 指令将 AX 中的数乘 10，再将 AX 和 BX 中的数相加，然后将结果保存到 BX 中，再等待下一次的输入，重复这个过程可以实现将输入的 ASCII 字符转换为一个多位十进制数字。		

```
-g9

AX=076A  BX=0000  CX=0182  DX=0000  SP=FFFC  BP=0000  SI=0000  DI=0000
DS=076A  ES=075A  SS=0769  CS=0772  IP=0009   NV UP EI PL ZR NA PE NC
0772:0009 8D160000      LEA     DX,[0000]                      DS:0000=3D4
-d0
076A:0000  4E 3D 3F 0A 0D 24 57 68-61 74 20 69 73 20 74 68   N=?..$What is th
076A:0010  65 20 6E 61 6D 65 20 6F-66 20 73 70 69 6E 64 6C   e name of spindl
076A:0020  65 20 58 20 3F 0A 0D 24-57 68 61 74 20 69 73 20   e X ?..$What is
076A:0030  74 68 65 20 6E 61 6D 65-20 6F 66 20 73 70 69 6E   the name of spin
076A:0040  64 6C 65 20 59 20 3F 0A-0D 24 57 68 61 74 20 69   dle Y ?..$What i
076A:0050  73 20 74 68 65 20 6E 61-6D 65 20 6F 66 20 73 70   s the name of sp
076A:0060  69 6E 64 6C 65 20 5A 20-3F 0A 0D 24 00 00 10 27   indle Z ?..$...'
076A:0070  E8 03 64 00 0A 00 01 00-00 00 00 00 00 00 00 00   ..d.............
-
```

```
-u
0772:009A BB0000        MOV     BX,0000
0772:009D B401          MOV     AH,01
0772:009F CD21          INT     21
0772:00A1 2C30          SUB     AL,30
0772:00A3 7C10          JL      00B5
0772:00A5 3C09          CMP     AL,09
0772:00A7 7F0C          JG      00B5
0772:00A9 98            CBW
0772:00AA 93            XCHG    BX,AX
0772:00AB B90A00        MOV     CX,000A
0772:00AE F7E1          MUL     CX
0772:00B0 93            XCHG    BX,AX
0772:00B1 03D8          ADD     BX,AX
0772:00B3 EBE8          JMP     009D
0772:00B5 C3            RET
```

返回后会调用过程 crlf，这里通过调用 INT 21H 中的过程 02H 来将 0DH 和 0AH 依次输出，用于换行。

```
-u
0772:00F5 B20A          MOV     DL,0A
0772:00F7 B402          MOV     AH,02
0772:00F9 CD21          INT     21
0772:00FB B20D          MOV     DL,0D
0772:00FD B402          MOV     AH,02
0772:00FF CD21          INT     21
0772:0101 C3            RET
```

下面会调用 INT 21H 中的功能 09H 和功能 01H 来实现将内存中的字符串输出，然后再读取一个字符，作为 x 的标识符，重复这个过程得到 y 和 z 的标识符。

最后是处理 hanoi 的过程，这里在 BX 中存储参数 N 的值，这里会先

将 BX 和 1 比较，如果相等就直接输出 X 移动到 Z 的过程，

```
-u
0772:0059 83FB01        CMP     BX,+01
0772:005C 7417          JZ      0075
0772:005E E82B00        CALL    008C
0772:0061 4B            DEC     BX
0772:0062 87F7          XCHG    SI,DI
0772:0064 E8F2FF        CALL    0059
0772:0067 E82900        CALL    0093
0772:006A E80C00        CALL    0079
0772:006D 4B            DEC     BX
0772:006E 87CE          XCHG    CX,SI
0772:0070 E8E6FF        CALL    0059
0772:0073 EB03          JMP     0078
0772:0075 E80100        CALL    0079
0772:0078 C3            RET
```

这里会调用 save 过程来保存当前 N,X,Y,Z 中的信息，然后在 hanoi 过程中进行递归调用，此时需要将 N-1 个盘子先从 X 移动到 Y，因此参数中的 Y 和 Z 的位置交换，这里需要将寄存器 SI 和 DI 进行交换。

```
-u
0772:008D 53            PUSH    BX
0772:008E 51            PUSH    CX
0772:008F 56            PUSH    SI
0772:0090 57            PUSH    DI
0772:0091 55            PUSH    BP
0772:0092 C3            RET
```

进入到递归调用的部分，这里 BX 的值为 2，下面再按照同样的方法进行递归调用，这里也需要交换 SI 和 DI 的值，再次进行递归调用。

```
-t

AX=020D  BX=0002  CX=0078  DX=000D  SP=FFF0  BP=0061  SI=007A  DI=0079
DS=076A  ES=075A  SS=0769  CS=0772  IP=0059   NV UP EI PL NZ NA PO NC
0772:0059 83FB01        CMP     BX,+01
-u
0772:0059 83FB01        CMP     BX,+01
0772:005C 7417          JZ      0075
0772:005E E82B00        CALL    008C
0772:0061 4B            DEC     BX
0772:0062 87F7          XCHG    SI,DI
0772:0064 E8F2FF        CALL    0059
0772:0067 E82900        CALL    0093
0772:006A E80C00        CALL    0079
0772:006D 4B            DEC     BX
0772:006E 87CE          XCHG    CX,SI
0772:0070 E8E6FF        CALL    0059
0772:0073 EB03          JMP     0078
0772:0075 E80100        CALL    0079
0772:0078 C3            RET
```

当 BX 中的值为 1 时，就会调用 print 来实现将一步操作进行输出，而不再进行递归调用，下面是 print 过程，这里分别将 CX 中的值，N 的值和 DI 的值进行输出。

```
-u
0772:0079 8BD1          MOV     DX,CX
0772:007B B402          MOV     AH,02
0772:007D CD21          INT     21
0772:007F E83400        CALL    00B6
0772:0082 8BD7          MOV     DX,DI
0772:0084 B402          MOV     AH,02
0772:0086 CD21          INT     21
0772:0088 E86A00        CALL    00F5
0772:008B C3            RET
```

在输出 N 的过程中，还需要将当前 BX 中的值转换为 ASCII，由于 N 转换成十进制后的结果可能不止一位，因此，这里需要调用 binidec 过程来转换为十进制形式的字符串并输出。

在转换为十进制字符的过程中，这里需要先从内存中得到除数，这个除数为 10000 到 1，用于得到十进制中的每个位，然后将除法运算完的结果存储到 DL 中，然后判断内存中的变量 flag 是否为 0，如果是 0，就说明当前还没有出现非 0 的数字，需要在后面判断是否为 0 才能转换为

ASCII 字符，否则会出现先导 0. 这里先将 N 的值除以 10000，因此得到的结果为 0，此时为先导 0，不能输出，直接进入到下一轮循环，将除数改为 1000。后面执行的操作同理。

```
AX=0000  BX=0001  CX=0005  DX=0000  SP=FFDA  BP=0061  SI=006E  DI=007A
DS=076A  ES=075A  SS=0769  CS=0772  IP=00D2   NV UP EI PL ZR NA PE NC
0772:00D2 833E6C0000    CMP     WORD PTR [006C],+00              DS:00
-t

AX=0000  BX=0001  CX=0005  DX=0000  SP=FFDA  BP=0061  SI=006E  DI=007A
DS=076A  ES=075A  SS=0769  CS=0772  IP=00D7   NV UP EI PL ZR NA PE NC
0772:00D7 750B          JNZ     00E4
-t

AX=0000  BX=0001  CX=0005  DX=0000  SP=FFDA  BP=0061  SI=006E  DI=007A
DS=076A  ES=075A  SS=0769  CS=0772  IP=00D9   NV UP EI PL ZR NA PE NC
0772:00D9 80FA00        CMP     DL,00
```

```
AX=0000  BX=0001  CX=0005  DX=0000  SP=FFDA  BP=0061  SI=006E  DI=007A
DS=076A  ES=075A  SS=0769  CS=0772  IP=00DC   NV UP EI PL ZR NA PE NC
0772:00DC 740D          JZ      00EB
-t

AX=0000  BX=0001  CX=0005  DX=0000  SP=FFDA  BP=0061  SI=006E  DI=007A
DS=076A  ES=075A  SS=0769  CS=0772  IP=00EB   NV UP EI PL ZR NA PE NC
0772:00EB 83C602        ADD     SI,+02
-t

AX=0000  BX=0001  CX=0005  DX=0000  SP=FFDA  BP=0061  SI=0070  DI=007A
DS=076A  ES=075A  SS=0769  CS=0772  IP=00EE   NV UP EI PL NZ AC PO NC
0772:00EE E2D7          LOOP    00C7
```

由于这里 N 的值为 1，只有到除数为 1 时，得到的计算结果是非 0 的，此时会将 1 转换为 ASCII 字符，然后通过调用 INT 21H 中的功能 02H 将这个字符进行输出。

```
AX=0001  BX=0001  CX=0001  DX=0000  SP=FFDA  BP=0061  SI=0076  DI=007A
DS=076A  ES=075A  SS=0769  CS=0772  IP=00CC   NV UP EI PL NZ NA PO NC
0772:00CC F734          DIV     WORD PTR [SI]                    DS:0076=0001
-t

AX=0001  BX=0001  CX=0001  DX=0000  SP=FFDA  BP=0061  SI=0076  DI=007A
DS=076A  ES=075A  SS=0769  CS=0772  IP=00CE   NV UP EI PL NZ NA PO NC
0772:00CE 8BDA          MOV     BX,DX
-t

AX=0001  BX=0000  CX=0001  DX=0000  SP=FFDA  BP=0061  SI=0076  DI=007A
DS=076A  ES=075A  SS=0769  CS=0772  IP=00D0   NV UP EI PL NZ NA PO NC
0772:00D0 8AD0          MOV     DL,AL
```

```
AX=0001  BX=0000  CX=0001  DX=0031  SP=FFDA  BP=0061  SI=0076  DI=007A
DS=076A  ES=075A  SS=0769  CS=0772  IP=00E7   NV UP EI PL NZ NA PO NC
0772:00E7 B402          MOV     AH,02
-t

AX=0201  BX=0000  CX=0001  DX=0031  SP=FFDA  BP=0061  SI=0076  DI=007A
DS=076A  ES=075A  SS=0769  CS=0772  IP=00E9   NV UP EI PL NZ NA PO NC
0772:00E9 CD21          INT     21
-t

AX=0201  BX=0000  CX=0001  DX=0031  SP=FFD4  BP=0061  SI=0076  DI=007A
DS=076A  ES=075A  SS=0769  CS=F000  IP=14A0   NV UP DI PL NZ NA PO NC
F000:14A0 FB            STI
-t

AX=0201  BX=0000  CX=0001  DX=0031  SP=FFD4  BP=0061  SI=0076  DI=007A
DS=076A  ES=075A  SS=0769  CS=F000  IP=14A1   NV UP EI PL NZ NA PO NC
F000:14A1 FE38          ???     [BX+SI]                         DS:0076=01
-t
1
AX=0231  BX=0000  CX=0001  DX=0031  SP=FFD4  BP=0061  SI=0076  DI=007A
DS=076A  ES=075A  SS=0769  CS=F000  IP=14A5   NV UP EI PL NZ NA PO NC
F000:14A5 CF            IRET
```

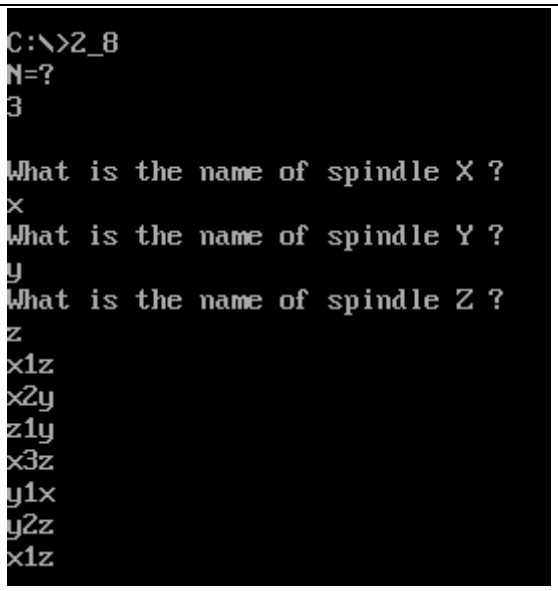
当输出执行完成后，将返回到 N 为 2 时调用的 Hanoi 过程，然后调用 restor 过程实现将寄存器恢复。下面再调用依次 print 过程，此时会将第二步的移动情况进行输出。

```
-g6d
x2y

AX=020D  BX=0002  CX=0078  DX=000D  SP=FFF0  BP=006A  SI=007A  DI=0079
DS=076A  ES=075A  SS=0769  CS=0772  IP=006D   NV UP EI PL NZ NA PE NC
0772:006D 4B            DEC     BX
```

下面再将 CX 和 SI 进行交换，表示将当前这一次调用时的 X 和 Y 进行交换，即通过子过程将 N-1 个盘子从 Y 移动到 Z，然后再进行一次递归调用。

下面是运行这个程序后的完整的执行结果。



下面增加对文件的读写。

这里需要增加下面的数据段定义，这里增加了字符串变量来存储文件名，并增加输入和输出到文件的缓冲区，以及打开文件后的句柄。

```
prompt      db      'File ?',0dh,0ah,'$'
input_name  db      'input.txt',0
output_name db      'output.txt',0
iHandle     dw      ?
oHandle     dw      ?
ibuf        db      5 dup(?)
obuf        db      5 dup(?)
```

下面再通过一个字符的输入来判断是否要读取文件，这里选择在输入为空格时选择从文件进行读写，从文件读取输入的部分调用 get_file 过程。

```
mov     ah,01h
int     21h
cmp     al,20h
jne     get_input                       ;不从文件输入
call    get_file
call    hanoi
ret
```

这里需要通过 INT 21H 中的功能 3DH 来打开特定的文件，然后从文件中读取 4 个字符，分别代表 N，X，Y，Z 的值，其中 N 需要进行转换，这

里实验 xor 指令来将 ASCII 转换为数字。然后依次将这些值存储到寄存器中，作为参数。

```
;从文件获取N X Y Z的值
    mov     ah,3dh
    mov     al,2
    lea     dx,input_name
    int     21h
    mov     iHandle,ax

    mov     ah,3fh
    mov     cx,4
    mov     bx,iHandle
    lea     dx,ibuf
    int     21h
    ;将缓冲区的内容转换为参数
    mov     bx,1
    mov     cl,ibuf[bx]
    xor     ax,ax
    mov     al,ibuf[bx + 1]
    mov     si,ax
    mov     al,ibuf[bx + 2]
    mov     di,ax
    ;close file
    mov     ah,3eh
    mov     bx,iHandle
    int     21h
    ;设置N的值
    xor     bx,bx
    mov     bl,ibuf[bx]
    xor     bx,30h
    ret
get_file    endp
```

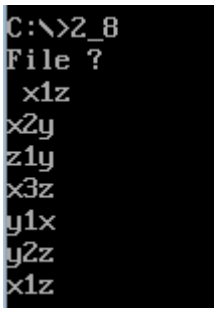
下面再实现将输出写到 output.txt 中。这里定义了过程 write，首先需要将寄存器的内容进行保存，然后将 x，n，z 的内容写到缓冲区中，再打开文件，将这些内容写入，最后再将寄存器恢复。

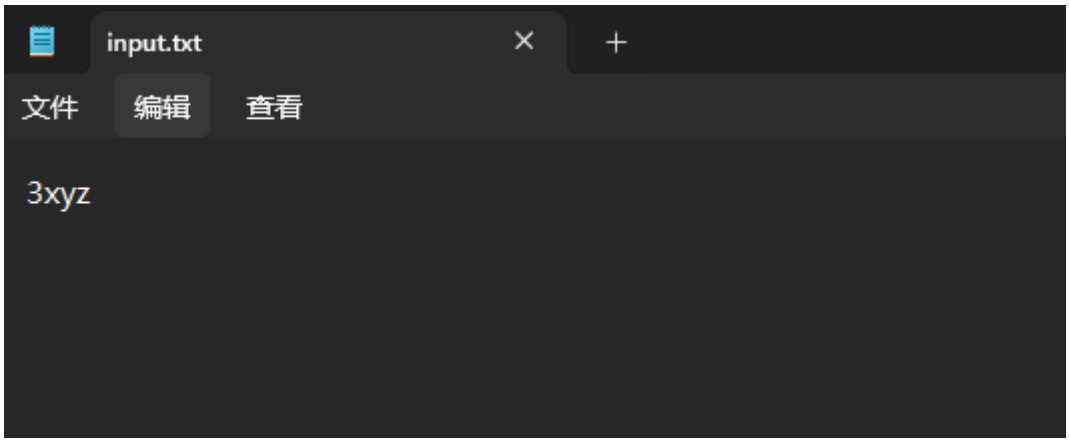
```
;--------------------------------
write   proc    near
    push    bx
    push    cx
    push    si
    push    di
;set buffer
    mov     ax,bx
    xor     bx,bx
    mov     obuf[bx],cl
    mov     cx,di
    mov     obuf[bx+2],cl
    xor     ax,30h
    mov     obuf[bx+1],al
    mov     obuf[bx+3],0dh
    mov     obuf[bx+4],0ah
    ;打开文件
    mov     ah,3dh
    mov     al,2
    lea     dx,output_name
    int     21h
    mov     oHandle,ax
    ;写文件
    mov     ah,40h
    mov     cx,5
    mov     bx,oHandle
    lea     dx,obuf
    int     21h


    pop     di
    pop     si
    pop     cx
    pop     bx
    ret
write   endp
```

带有文件的运行结果如图。

这里输入了一个空格，然后读取文件中的内容，这里 N X Y Z 的值分别为 3，x，y，z，然后将内容写入到文件中，如图。




input.txt
文件
编辑
查看
3xyz


input.txt
output.txt
文件
编辑
查看
x1z
x2y
z1y
x3z
y1x
y2z
x1z

问题及收获：

1. 这次实验学会了汇编中实现递归的方式，这里在 hanoi 问题中使用递归。
2. 这里学会了使用寄存器进行传参的方法，其中，参数包含 N,X,Y,Z，将这些值分别存入到 BX,CX,SI,DI 中。
3. 三种入参和出参使用的方法中，这里主要使用了寄存器法，寄存器法可以在参数较少的时候使用，使用方便，缺点是寄存器数量有限，因此在参数较多是不方便使用。

   约定内存法可以支持更多的参数的情况，不占用寄存器，缺点是需要有额外的变量定义，不方便使用。

   堆栈法支持参数较多，存在递归调用的情况，不占用寄存器，也不需要额外的变量进行定义。缺点是需要注意出栈和入栈的次序，且与子程序混在一起，可能会混淆。