1. **程序设计要求：文档比较**
   1. 输入：一个文件夹，仅包含N个ASM源代码
   2. 输出：一个N\*N的矩阵M，写入一个TXT文件，文件名output.txt， M(i,j) = 源代码i和和源代码j的文档相似度
2. **设计思路：**
   1. 整体结构：通过用户输入目录地址（如d:\codes）来打开asm文件的目录；然后程序获取文件夹内的所有asm文件的文件名并存入缓冲区；然后做n(n-1)/2次比较，得到每个文件之间的相似度，并将相似度存入缓冲区；最后将缓冲区的相似度数据通过矩阵形式输出到d:\output.txt文件内。
   2. 相似度计算：相似度计算采用最小编辑距离算法。即：计算两个字符串之间相互转换最少需要多少次。我们对字符可能进行的操作有三种：将s[1…n]转换为t[1…m]当然需要将所有的s转换为所有的t，所以，d[n,m]就是我们所需的结果。如果我们可以使用k个操作数把s[1…i]转换为t[1…j-1]，我们只需要把t[j]加在最后面就能将s[1…i]转换为t[1…j]，操作数为k+1；如果我们可以使用k个操作数把s[1…i-1]转换为t[1…j]，我们只需要把s[i]从最后删除就可以完成转换，操作数为k+1；如果我们可以使用k个操作数把s[1…i-1]转换为t[1…j-1]，我们只需要在需要的情况下（s[i] != t[j]）把s[i]替换为t[j]，所需的操作数为k+cost（cost代表是否需要转换，如果s[i]==t[j]，则cost为0，否则为1）。
   3. 文本过滤：由于文件含有大量影响相似度计算的其他字符，所以在读取文件的时候，先进行文本过滤，只保留大小写字母和数字，将汉字、符号、空格、换行去掉，使得结果更加准确。
   4. 压缩状态节省空间：由算法我们需要将过滤后的文本存入缓冲区。由于内存限制为1MB，两个文件的缓冲区各自设置为7000字节。并且，之后需要一个相似度比较矩阵，若过滤后缓冲区1和缓冲区2的长度分别为n和m，本应是需要(n+1)\*(m+1)字节的空间进行状态转移，此时远超1MB内存。此时就必须要进行状态压缩。通过处理，将空间压缩为3\*(m+1)行。其实还可以进一步压缩为3\*3行，但此次时间有限，就采用压缩为3\*(m+1)行的策略，以后可以改进。
   5. 变换为矩阵格式：将缓冲区的连续字符，变换为矩阵格式。因为这里是一个对称矩阵，所以这里我们为了方便处理，输出的为上三角矩阵。读取文件名时可以得到文件的数量，知道文件数量之后就可以得知每一行的数据的个数依次为:n-1,n-2,…,1个。按照此方法，读取缓冲区，在特定位置加上换行与若干空格即可。
3. **核心代码**
   1. 相似度计算：

;\*\*\*相似度计算\*\*\*

SAME: MOV SI, 1 ;文件1的指针

MOV DI, 1 ;文件2的指针

FOR1: CMP SI, 1

JE FLINE ;第一特行殊处理

XOR DX, DX

MOV AX, SI ;做除法，根据余数确定实际空间行号

MOV BX, 2

DIV BX

CMP DX, 0

JZ EVENN

;寄存器的值对应的意义：AH->left(左边的行号) AL->up(上边的行号) DH->leftUp(左上方的行号) DL->ii(当前行号)

ODDN: MOV AH, 1 ;奇数行，对应到实际空间为第1行

MOV DH, 2

MOV AL, 2

MOV DL, 1

JMP DP

EVENN: MOV AH, 2 ;偶数行，对应到实际空间为第2行

MOV DH, 1

MOV AL, 1

MOV DL, 2

JMP DP

FLINE: MOV AH, 1

MOV DH, 0

MOV AL, 0

MOV DL, 1

DP: LEA BX, FILE1

DEC BX

MOV CH, [BX + SI] ;BX减了一下，实际为[BX + SI - 1]

LEA BX, FILE2

DEC BX

MOV CL, [BX + DI] BX减了一下，实际为[BX + DI - 1]

CMP CH, CL

JE LEFTUP

;当前字符不等，那么为左边或者上边

;算左边

PUSH DX

PUSH AX

;MOV CL, 4

;SHR AX, CL ;AX逻辑右移4位，使得AX是AH(左边的行号)

MOV AL, AH

XOR AH, AH

MOV CX, FILE2L

INC CX ;每一行的长度是文件2的长度+1

MUL CX

MOV BX, AX

ADD BX, DI

DEC BX

ADD BX, BX

MOV DX, MATRIX[BX] ;BX减了一下，实际为[(BX + DI - 1)\*2],左方数据

;算上边

POP AX

PUSH DX ;存入左方数据

XOR AH, AH ;AX高四位置零，使得AX为AL(上方行号)

MUL CX

MOV BX, AX

ADD BX, DI

ADD BX, BX

MOV CX, MATRIX[BX] ;上方数据([(BX + DI)\*2])

POP DX ;取出左方数据

CMP DX, CX

JA MAXN

JMP THEMAX

FOR2: JMP FOR1

MAXN: MOV CX, DX ;上边更大

;当前位置放入更大的数

THEMAX: POP DX

PUSH CX

XOR DH, DH ;取得当前行

MOV AX, DX

MOV CX, FILE2L

INC CX

MUL CX

MOV BX, AX

ADD BX, DI

ADD BX, BX

POP CX

MOV MATRIX[BX], CX

JMP NEXT

;左上方

LEFTUP: PUSH DX

MOV DL, DH

XOR DH, DH ;使得DX为DH

MOV AX, DX

MOV CX, FILE2L

INC CX ;一行的个数是文件2的长度+1

MUL CX ;AX中的行号乘以CX中的每行的元素个数 结果在AX中

MOV BX, AX ;结果存入基址寄存器BX

ADD BX, DI

DEC BX

ADD BX, BX

MOV CX, MATRIX[BX] ;BX减了，实际为[(BX + DI - 1)\*2]

INC CX ;此时CX是matrix[leftUp][j - 1] + 1

POP DX

PUSH CX

XOR DH, DH ;DX高4位置零，使得结果为DL，当前行号

MOV AX, DX

MOV CX, FILE2L

INC CX ;一行的个数是文件2的长度+1

MUL CX ;AX中的行号乘以CX中的每行的元素个数 结果在AX中

MOV BX, AX ;计算的是当前行对应的结果

ADD BX, DI

ADD BX, BX

POP CX

MOV MATRIX[BX], CX

JMP NEXT

NEXT: INC DI

MOV AX, FILE2L

CMP DI, AX

JBE FOR2

MOV DI, 1

INC SI

MOV AX, FILE1L

CMP SI, AX

JBE FOR2

;求结果存放的行号(1或者2)

XOR DX, DX

MOV AX, FILE1L

MOV BX, 2

DIV BX

CMP DX, 0

JE EVEANS

ODDANS: MOV SI, 1 ;奇数行

JMP ANS

EVEANS: MOV SI, 2 ;偶数行

;得到答案

ANS: MOV DI, FILE2L

MOV AX, SI

MOV CX, FILE2L

INC CX

MUL CX

MOV BX, AX

ADD BX, DI

ADD BX, BX

MOV AX, MATRIX[BX]

MOV TMPLEN, AX

;\*\*\*求百分比\*\*\*

MOV DI, ANSSETP

PERC: XOR DX, DX

MOV AX, TMPLEN

MOV CX, 100

MUL CX ;相同长度先乘100,再除以最大长度

MOV CX, MAXLEN

DIV CX ;此时百分比在AX中 最大为100，所以结果其实在AL中

CMP AX, 100

JE HUND

MOV BL, 10 ;除以10，得到十位和个位

DIV BL

CMP AL, 0

JE NOTEN

ADD AL, 30H

MOV ANSSET[DI], AL

INC DI

NOTEN: ADD AH, 30H

MOV ANSSET[DI], AH

INC DI

JMP OVER

HUND: MOV ANSSET[DI], 31H

INC DI

MOV ANSSET[DI], 30H

INC DI

MOV ANSSET[DI], 30H

INC DI

OVER: MOV ANSSET[DI], 37

INC DI

MOV ANSSET[DI], 00H

INC DI

MOV ANSSETP, DI

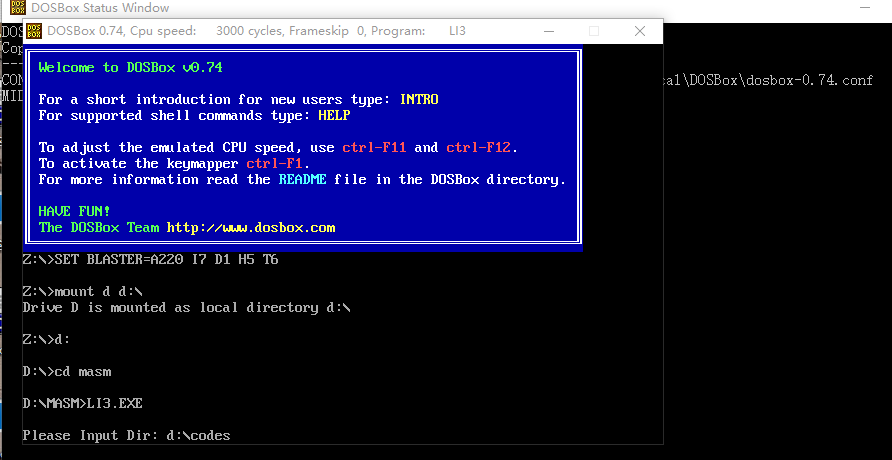
MOV AL, WHICHF1

MOV AH, WHICHF2

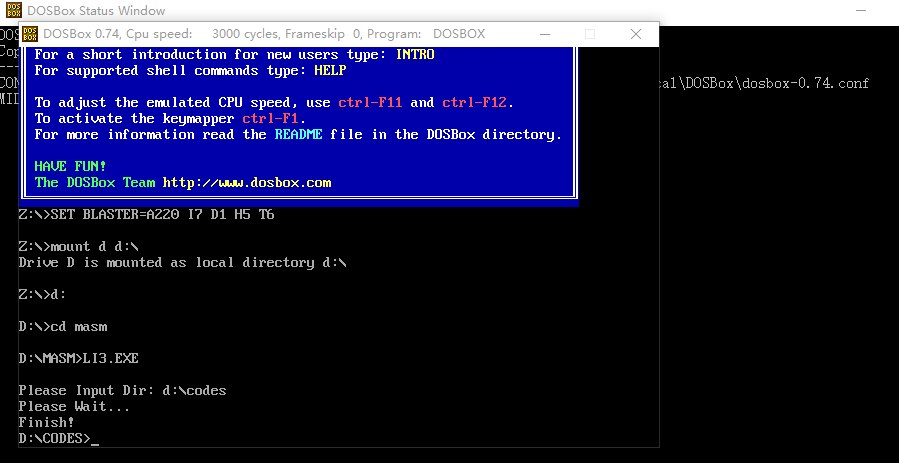
INC AH

JMP CHOIE1

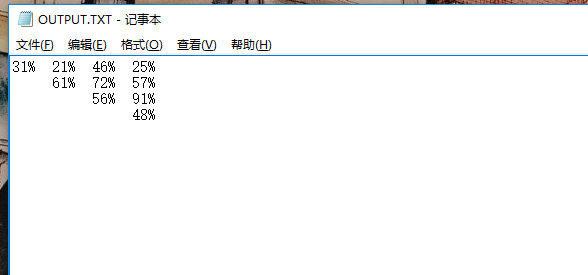
1. **运行调试结果**
   1. 编译运行，输入目录。如d:\codes



* 1. 此时显示Please Wait...
  2. 当显示Finish!时，表示成功。



* 1. 进入到d:\output.txt,查看结果。结果为上三角矩阵（不含对角线），含义与题目一样。此时(1,1)表示，第1个文件与第2个文件的相似度。(i,j)表示第i个文件与第j+1文件的相似度。



1. **体会与收获**

学会了最基本的文档比较算法，最小编辑距离算法。学会了汇编语言的基础运用，以及部分DOS系统功能调用。之前写的都是高级语言，能够方便的调试，逻辑也非常清楚。但是写汇编的时候，调试非常麻烦，不是很直观。后来慢慢习惯。有时候又会遇到很玄学的问题，比如莫名其妙内存就超了，所以如果文件超过800行代码，可能相似度计算精度会有所下降。然后就是很多时候都在不断地改BUG，半小时写代码，两小时调BUG。不管怎样，这次汇编的课程设计还是收获挺多的，提高了自己逻辑思维能力，让自己更加耐心。