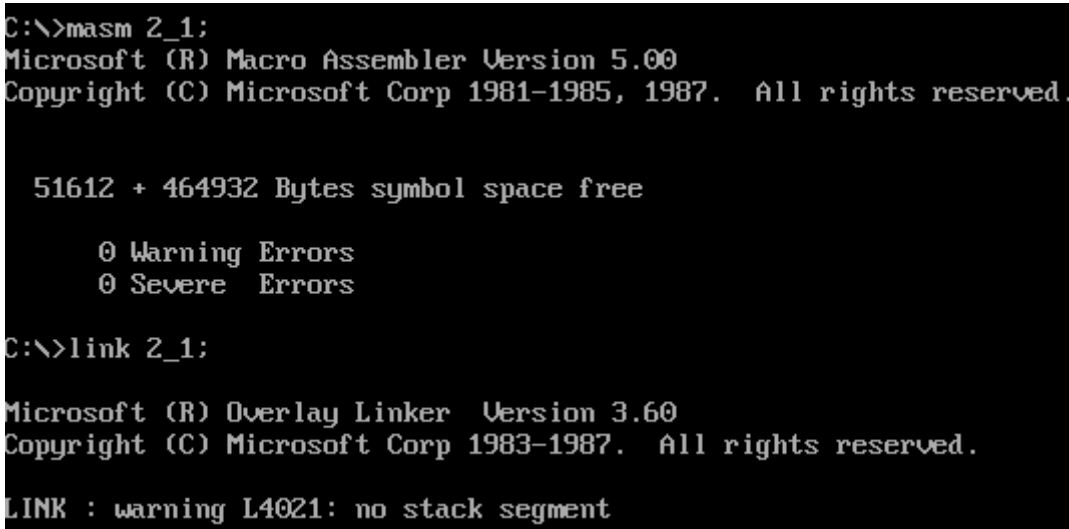


山东大学 计算机科学与技术 学院

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 3：例 2.1，2.3		
实验学时：2	实验日期：20241028	
实验目的：继续熟悉 MASM、LINK、DEBUG、EDIT、TD 等汇编工具。掌握汇编语言循环程序与分支程序的设计思路。掌握示例程序中的寻址方式、常量的含义，及各个伪指令。了解字符串在内存中的存储方式，存储形式 DW、DB 的不同。		
实验环境：Windows11、DOSBox-0.74、Masm64		
源程序清单： 1. 2_1.asm （示例 2.1） 2. 2_3.asm （示例 2.3）		
编译及运行结果： 1. 例 2.1 程序的编译和运行结果如图。 		

```

C:\>2_1
stock number?
05 Excavators
stock number?
09 Presses
stock number?
23 Processors
stock number?
Not in table!
C:\>a_

```

下面对这个程序进行调试。这里首先实现了输出‘stock number?’并接收输入，这里输入的是一个两位数字。然后将输入的内容分别存储到 AX 的高位和低位，这里用 ASCII 表示。

这里使用了系统调用 INT 21H，其中 09H 用于输出字符串，0AH 实现将键盘输入到缓冲区。

```

0773:0022 A01200      MOV     AL,[0012]
0773:0025 8A261300      MOV     AH,[0013]
0773:0029 B90600      MOV     CX,0006
0773:002C 8D361500      LEA     SI,[0015]
0773:0030 3B04          CMP     AX,[SI]
0773:0032 7410          JZ      0044
0773:0034 83C60E      ADD     SI,+0E
0773:0037 E2F7      LOOP   0030
0773:0039 8D167A00      LEA     DX,[007A]
0773:003D B409      MOV     AH,09
0773:003F CD21      INT     21
0773:0041 EB14      JMP     0057

```

然后读取内容中存储的表的内容，并将地址存储到 SI 中。

```

-d15
076A:0010          30 35 20 45 78 63 61 76 61 74 6F          05 Excavato
076A:0020 72 73 20 30 38 20 4C 69-66 74 65 72 73 20 20 20 rs 08 Lifters
076A:0030 20 30 39 20 50 72 65 73-73 65 73 20 20 20 20 31 09 Presses 1
076A:0040 32 20 56 61 6C 76 65 73-20 20 20 20 20 32 33 20 2 Valves 23
076A:0050 50 72 6F 63 65 73 73 6F-72 73 20 32 37 20 50 75 Processors 27 Pu
076A:0060 6D 70 73 20 20 20 20 20-20 20 20 20 20 20 20 20 mps
076A:0070 20 20 20 20 20 20 20 0D-0A 24 4E 6F 74 20 69 6E ..$Not in
076A:0080 20 74 61 62 6C 65 21 24-00 00 00 00 00 00 00 00 table!$.
076A:0090 1F 2B 60 50 88

```

下面通过一个循环来依次查找与输入对应的编号，直到查找到或者搜索完整个表。当没有搜索到的时候，就将 SI 的值加 14，并在新的地址下继续与输入的内容进行比较。

```

-u
0773:0030 3B04      CMP     AX,[SI]
0773:0032 7410      JZ      0044
0773:0034 83C60E     ADD     SI,+0E
0773:0037 E2F7      LOOP   0030
0773:0039 8D167A00   LEA     DX,[007A]
0773:003D B409      MOV     AH,09
0773:003F CD21      INT     21
0773:0041 EB14      JMP     0057
0773:0043 90         NOP
0773:0044 B90700     MOV     CX,0007
0773:0047 8D3E6900   LEA     DI,[0069]
0773:004B F3         REPZ
0773:004C A5         MOUSW
0773:004D 8D166900   LEA     DX,[0069]

```

这里需要确保源代码中定义的数据段部分的字符串占用 12 个字节，此时再加上编号部分刚好 14 个字节才能确保答案正确，否则输出的结果除了第一个外都会出现异常。

如果搜索到对应的编号，就会跳转到 A30 代码段，此时会根据当前的 SI 的地址来找到要存入的字符串，并依次读取 14 个字符，然后将内容进行输出。

```

-u
0773:0044 B90700     MOV     CX,0007
0773:0047 8D3E6900   LEA     DI,[0069]
0773:004B F3         REPZ
0773:004C A5         MOUSW
0773:004D 8D166900   LEA     DX,[0069]
0773:0051 B409      MOV     AH,09
0773:0053 CD21      INT     21
0773:0055 EBB4      JMP     000B

```

如果没有找到对应的编号，就需要在 6 次循环结束后，将 mess 中的内容输出，即 Not in table。

2. 例 2.3

对程序进行调试。

这里先设置了 DS 的值，然后依次将 S5, S6, S7, S8, S9, S10 的值设置为 0。

```

C:\>debug 2_3.exe
-u
076C:0000 1E         PUSH    DS
076C:0001 2BC0      SUB     AX,AX
076C:0003 50        PUSH    AX
076C:0004 B86A07     MOV     AX,076A
076C:0007 8ED8      MOV     DS,AX
076C:0009 C70614000000 MOV     WORD PTR [0014],0000
076C:000F C70616000000 MOV     WORD PTR [0016],0000
076C:0015 C70618000000 MOV     WORD PTR [0018],0000
076C:001B C7061A000000 MOV     WORD PTR [001A],0000

```

设置完成 DS 的值后数据段的内容如图。其中 S5-S10 存储在内存中的 14-1E 的位置，由于在数据段的定义中，这些值已经为 0，这里执行完将 0 给 S5-10 赋值后内存中的内容没有变化。

```

-D0
076A:0000 38 00 45 00 54 00 52 00-49 00 58 00 63 00 3F 00 8.E.T.R.I.X.c.?
076A:0010 64 00 50 00 00 00 00 00-00 00 00 00 00 00 00 00 d.P.....
076A:0020 1E 2B C0 50 B8 6A 07 8E-D8 C7 06 14 00 00 00 C7 .+.P.j.....
076A:0030 06 16 00 00 00 C7 06 18-00 00 00 C7 06 1A 00 00 .....
076A:0040 00 C7 06 1C 00 00 00 C7-06 1E 00 00 00 B9 0A 00 .....
076A:0050 BB 00 00 8B 07 3D 3C 00-7C 32 3D 46 00 7C 27 3D .....=<.12=F.1'
076A:0060 50 00 7C 1C 3D 5A 00 7C-11 3D 64 00 75 06 FF 06 P.1.=Z.1.=d.u...
076A:0070 1E 00 EB 1C FF 06 1C 00-EB 16 FF 06 1A 00 EB 10 .....

```

下面在 CX 和 BX 的位置分别存储循环的计数和 grade 在内存中的首地址。

```

AX=076A BX=0000 CX=0096 DX=0000 SP=FFFC BP=0000 SI=0000 DI=0000
DS=076A ES=075A SS=0769 CS=076C IP=002D NU UP EI PL ZR NA PE NC
076C:002D B90A00 MOV CX,000A
-T
AX=076A BX=0000 CX=000A DX=0000 SP=FFFC BP=0000 SI=0000 DI=0000
DS=076A ES=075A SS=0769 CS=076C IP=0030 NU UP EI PL ZR NA PE NC
076C:0030 BB0000 MOV BX,0000

```

后面依次读取数据，并将分数进行比较，如这里是第一个值，这个值为 56，会跳转到 five 对应的代码，即将 s5 的值加 1，然后将 BX 的值加 2，再用 loop 指令跳转到 compare 部分的代码。

```

AX=003B BX=0000 CX=000A DX=0000 SP=FFFC BP=0000 SI=0000 DI=0000
DS=076A ES=075A SS=0769 CS=076C IP=006C NU UP EI NG NZ AC PE CY
076C:006C FF061400 INC WORD PTR [0014] DS:0014=0000
-U
076C:006C FF061400 INC WORD PTR [0014]
076C:0070 83C302 ADD BX,+02
076C:0073 E2BE LOOP 0033
076C:0075 CB RETF
076C:0076 83C404 ADD SP,+04
076C:0079 0BC0 OR AX,AX
076C:007B 7503 JNZ 0080
076C:007D E9A500 JMP 0125
076C:0080 C7867AFF0000 MOV WORD PTR [BP+FF7A],0000
076C:0086 EB04 JMP 008C
076C:0088 FF867AFF INC WORD PTR [BP+FF7A]

```

执行完前面的一轮循环后，S5 对应位置的值为 1，如下图。

```

-D14
076A:0010 01 00 00 00-00 00 00 00 00 00 00 00 00 00 00
076A:0020 1E 2B C0 50 B8 6A 07 8E-D8 C7 06 14 00 00 00 C7
076A:0030 06 16 00 00 00 C7 06 18-00 00 00 C7 06 1A 00 00
076A:0040 00 C7 06 1C 00 00 00 C7-06 1E 00 00 00 B9 0A 00
076A:0050 BB 00 00 8B 07 3D 3C 00-7C 32 3D 46 00 7C 27 3D
076A:0060 50 00 7C 1C 3D 5A 00 7C-11 3D 64 00 75 06 FF 06
076A:0070 1E 00 EB 1C FF 06 1C 00-EB 16 FF 06 1A 00 EB 10
076A:0080 FF 06 18 00 EB 0A FF 06-16 00 EB 04 FF 06 14 00
076A:0090 83 C3 02 E2

```

当执行完所有的循环后，内存的值如下图。其中地址 14, 16, 18, 1A, 1C, 1E 分别对应 S5-S10，因此这些值分别为 1, 2, 1, 4, 1, 1，与数据段定义中给出的 10 个成绩的值对应。

```

-D14
076A:0010      01 00 02 00-01 00 04 00 01 00 01 00
076A:0020  1E 2B C0 50 B8 6A 07 8E-D8 C7 06 14 00 00 00 C7
076A:0030  06 16 00 00 00 C7 06 18-00 00 00 C7 06 1A 00 00
076A:0040  00 C7 06 1C 00 00 00 C7-06 1E 00 00 00 B9 0A 00
076A:0050  BB 00 00 8B 07 3D 3C 00-7C 32 3D 46 00 7C 27 3D
076A:0060  50 00 7C 1C 3D 5A 00 7C-11 3D 64 00 75 06 FF 06
076A:0070  1E 00 EB 1C FF 06 1C 00-EB 16 FF 06 1A 00 EB 10
076A:0080  FF 06 18 00 EB 0A FF 06-16 00 EB 04 FF 06 14 00
076A:0090  83 C3 02 E2

```

问题及收获：

1. 这次实验中了解循环结构的实现方式。
2. 在例 2.1 中,在数据段的定义中的空格数量需要与代码中每一次循环的偏移量相对应,否则会造成结果出现异常,由于这里每次将 SI 的值加 14,因此需要将数据段定义中每一行的编码值及对应的字符串的长度之和为 14,即第二个字符串中补上的空格的数量应该根据单词的长度来确定。
3. 例 2.3 中通过 debug 工具来实现对内存的查看,并在这个过程中通过调试了解的多个分支和跳转的实现方式。
4. 通过例 2.1 了解了 DOS 调用,使用 INT 21H 指令来实现调用 0ah 和 09h 分别实现输入和输出,并在数据段中定义变量来接收缓冲区的输入和输出。