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：202120130276	姓名：王云强	班级：21.2 班
实验题目：实验三：例 2.1，例 2.3		
实验学时：2	实验日期：2023.11.03	

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：Windows10、DOSBox-0.74、Masm64

源程序清单：

1. TABSRCH.ASM（示例 2.1 源程序）

2. result.ASM（示例 2.3 源程序）

编译及运行结果：

示例 2.1 编译结果：

```
C:\>masm TABSRCH
Microsoft (R) Macro Assembler Version 5.00
Copyright (C) Microsoft Corp 1981-1985, 1987.  All rights reserved.

Object filename [TABSRCH.OBJ]:
Source listing  [NUL.LST]:
Cross-reference [NUL.CRF]:

  51592 + 464952 Bytes symbol space free

      0 Warning Errors
      0 Severe  Errors

C:\>LINK TABSRCH

Microsoft (R) Overlay Linker  Version 3.60
Copyright (C) Microsoft Corp 1983-1987.  All rights reserved.

Run File [TABSRCH.EXE]:
List File [NUL.MAP]:
Libraries [.LIB]:
LINK : warning L4021: no stack segment
```

示例 2.1 运行结果：

```
C:\>TABSRCH
stock nember?
23 PROCESSORS
stock nember?
27 PUMPS
stock nember?
05 EXCAVATORS
stock nember?
09 PRESSES
stock nember?
08 LIFTERS
stock nember?
12 VALVES
stock nember?
```

示例 2.3 编译结果：

```
C:\>masm result
Microsoft (R) Macro Assembler Version 5.00
Copyright (C) Microsoft Corp 1981-1985, 1987.  All rights reserved.

Object filename [result.OBJ]:
Source listing  [NUL.LST]:
Cross-reference [NUL.CRF]:

  51670 + 464874 Bytes symbol space free

      0 Warning Errors
      0 Severe  Errors

C:\>link result

Microsoft (R) Overlay Linker  Version 3.60
Copyright (C) Microsoft Corp 1983-1987.  All rights reserved.

Run File [RESULT.EXE]:
List File [NUL.MAP]:
Libraries [.LIB]:
LINK : warning L4021: no stack segment
```

示例 2.3 调试结果

```
-u
076C:0021 C7061C000000  MOV     WORD PTR [001C],0000
076C:0027 C7061E000000  MOV     WORD PTR [001E],0000
076C:002D B90A00        MOV     CX,000A
076C:0030 BB0000        MOV     BX,0000
076C:0033 8B07          MOV     AX,[BX]
076C:0035 3D3C00        CMP     AX,003C
076C:0038 7C32          JL      006C
076C:003A 3D4600        CMP     AX,0046
076C:003D 7C27          JL      0066
076C:003F 3D5000        CMP     AX,0050
-u
076C:0042 7C1C          JL      0060
076C:0044 3D5A00        CMP     AX,005A
076C:0047 7C11          JL      005A
076C:0049 3D6400        CMP     AX,0064
076C:004C 7506          JNZ     0054
076C:004E FF061E00      INC     WORD PTR [001E]
076C:0052 EB1C          JMP     0070
076C:0054 FF061C00      INC     WORD PTR [001C]
076C:0058 EB16          JMP     0070
076C:005A FF061A00      INC     WORD PTR [001A]
076C:005E EB10          JMP     0070
076C:0060 FF061800      INC     WORD PTR [0018]
```

示例 2.3 运行结果（从 debug 知，从 14 号单元开始存放内容即各个得分对应的排名，每一个字(即四位十六进制数)对应一个分数段的得分个数：

```
-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.进一步熟悉了 DOSBox 下的 debug 的使用，进一步熟练了对通过调试对相应内存单元内容进行查看的操作。

2.对示例 2.1 代码的解读：

数据段是这样的：

MAX 代表最多输入位数，ACT 记录输入字符的个数，STOKN 来储存输入的内容，STOKTAB 储存对应的库存品编号和对应库存品名称，DESCRN 是输出内容的储存位置,14 代表 14 个字符。

代码段是这样的：

首先输出提示输入内容，之后检查输入位数，如果没输入则退出，否则将输入第一位放入 AX 寄存器低位，第二位放入 AX 寄存器高位，设置循环计数器 CX 为 6（为了配合后面的 loop 指令）。之后在每次循环中只将库存品信息（STOKTAB）中的第一个字与 AX 比较，转换一下意思就是，只比较 AX 寄存器的内容（输入的两个数字对应的 ASCII 码）和对应库存品的前两个字符（即库存品编号对应的 ASCII 码）。由于在库存品信息（STOKTAB）中，数字是以 ASCII 码形式存放的，所以直接可以将其与

输入信息作比较，如果比较成功转向 A30 部分，否则则继续循环比较，如果都不符合，则输出“NOT IN TABLE”退出。A30 部分是循环 7 次，向输出内容 DESCRN 中存放 7 个字内容，即对应的股票信息，之后再输出对应股票信息，退出。

3.对示例 2.3 代码的解读：

数据段：GRADE 数组存放成绩，S5-S10 存放着对应分数段的分数个数，初始都为 0。

代码段：首先让 BX 做数组下标，来遍历数组，CX 循环计数器初始为 10 代表遍历 10 个数。每次循环中，取数组对应数字放入 AX 中，依次与 60、70、80、90、100 进行比较，如果不满足对应分数段，则跳转到 相应的数字处理部分进行处理。如成绩 88 先与 60 比较，再与 70 比较， 再与 80 比较，再与 90 比较，此时 88 小于 90，会跳转到 EIGHT 部分， 使得 s8 自增，之后再跳转到 CHANGE_ADDR 中，使得 BX 下标加 2（即向后移动一个数字），通过 LOOP 循环（CX 此时自减，并判断是否为 0）回到 COMPARE 继续比较下一个数。

4.取数组首地址的方式

在示例 2.3 中，有这样一条语句，MOV BX,OFFSET GRADE，但是之前做作业的时候，发现取数组首地址的语句可以为 LEA BX GRADE。后经过测试发现两个语句的意思是一样的，都是通过将 GRADE（数组）的首单元的地址放入到 BX 寄存器中，之后再通过寄存器间接取址，来将数组中的

内容取出来。