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：实验二：示例 2.4、实例 2.5		
实验学时：2	实验日期：2023.10.27	
实验目的： 1.继续熟悉 MASM、LINK、DEBUG、EDIT、TD 等汇编工具。 2.掌握汇编语言分支程序的设计思路。 3.理解条件转移指令和无条件转移指令的机理(哪个是根据状态寄存器，哪个是位移，哪个是段首址加有效地址）。了解段内短转移、段内近转移、段间转移、直/间接等的含义及其跳转范围。		
实验环境：Windows10、DOSBox-0.74、Masm64		
源程序清单： 1. Month.ASM（示例 2.4 源程序） 2.sample.ASM（示例 2.5 源程序） 3. Error.ASM（示例 2.5 修改后的源程序）		
编译及运行结果： 示例 2.4:		

```
1
JAN
month?
2
FEB
month?
3
MAR
month?
4
APR
month?
5
MAY
month?
6
JUN
month?
7
JUL
month?
8
AUG
month?
```

```
8
AUG
month?
9
SEP
month?
10
OCT
month?
11
NOV
month?
12
DEC
month?
```

示例 2.5（源程序）：

```
-rax
AX FFFF
:1
-g45
Invalid function number

AX=0001  BX=0368  CX=0406  DX=0059  SP=FFFC  BP=0000  SI=076A  DI=0000
DS=076A  ES=075A  SS=0769  CS=07A6  IP=0045   NV UP EI PL NZ NA PO NC
07A6:0045 CB            RETF
-rip
IP 0045
:0
-rax
AX 0001
:0
-g45
Error code is not in valid range(1-83)

AX=0000  BX=0000  CX=0406  DX=0010  SP=FFF8  BP=0000  SI=076A  DI=0000
DS=076A  ES=075A  SS=0769  CS=07A6  IP=0045   NV UP EI PL ZR NA PE NC
07A6:0045 CB            RETF
```

示例 2.5（修改后的程序）：

问题及收获：

1. 通过本实验，熟悉了分支程序的设计流程及大概设计思路，对课本上的知识有了更深的了解。

2. 对示例 2.4 程序的解读：

首先是数据段，由于月份最多两位数，所以先设置了 THREE 来设置最大输入限制（示例中是 3 位，包括回车），之后 MONIN 用了 label 伪指令，为下一个变量提供了不同的名称和大小属性，共享了同一内存位置，之后 MAX 定义了最多输入的字符（基本在这里无用），ACT 记录了输入的数字的位数，MON 来存储输入的数字。最关键的是 ALFMON 部分，是作为缓冲区使用，中间的???可以容用其他字符代替，而???代表随机填入在这里。

接下里是程序段，最开始的是 INPUT MONTH 部分，主要功能输出提示输入的字段，以及显示输入，并且输出换行回车，最后比较一下输入了多少个字符，如果为 0，则退出程序，不然的话继续向下执行。紧接着就是需要判断输入了几位数字，如果输入了两位则额外处理一下，把高位（也就是十位数的数字）放入 AH，把低位放入 AL。之后与 3030H 进

行异或（因为输入的是 ASCII 码，只需要真实数值部分），判断是不是两位数，如果是两位数则额外将 AL 中的数字额外加 10。之后转向定位部分，首先先要确定之前存放月份对应英文的缩写的数组首地址，由于数组从 0 开始，因此要 AL 自减，以 3 字节为最小跨越单位，定位到目标月份地址，以循环的方式来将对应月份英文放到缓冲区（循环三次，因为英文 3 字节），最后输出出来即可。

对于程序 2.4 而言，有一个特殊的地方就是在输入两位数时，程序只会比较高位是不是 0，如果不是 0，则清空高位，默认按高位为 1 来进行定位。即只要是输入两位数，如 11,12,32,41，只要是 X1 或者是 X2 形式，程序都会只根据末尾的 1 和 2 来输出是 NOV 还是 DEC，而不是出现像其他的一样的乱码。

3.对示例 2.5 程序的解读：

在示例 2.5 源程序中，在之前的程序里这是没有输入交互的，程序是直接根据 AX 寄存器的值来判断是哪种错误类型，进行返回。所以要修改程序的话，需要做两件事（也可以理解为一件事）。①输入界面交互，让程序可以被输入②将输入的数从 ASCII 码形式（因为输入默认是 ASCII 码，但是程序判断的时候是按真实数字来判断的）转换为数字形式，并储存在 AX 寄存器中。

在这里利用了与 2.4 程序相同的输入交互，即采用 21H 中断的 0AH，来将输入内容放到缓冲区里。首先是将输入设定为最多 3 个字符，设置一个 ACT 来记录输入的字符数，NUM 来存放输入的数。

之后，先判断输入了几位数，如果是 1 位数，则将 AX 寄存器的高八位（AH）放 30H（即的 ASCII 码），低八位放输入的数字，将 AX 内容与 3030H 异或,这样 AX 就会存放输入数据对应的真实数字。如果是 2 位数，则先将十位数转换为真实数字，并放在 AL 中，与 BH（存放 10）相乘，答案放在 AX 中，之后将个位数放入 BL 中，BH 中放 30H，并将低位数转换成真实数字，放入 BX 中，之后将 AX 与 BX 相加，答案放入 AX 即可。这样就完成了转换。接下来就是原程序部分了，首先程序判断数字是否大于 83 或者是小于 0，如果大于 83 或小于 0 则输出异常输入的提示。

之后再判断是否大于 35，如果大于 35，那是否大于 79，如果大于 79 那就将 BX 装入 ERTAB2（80-83 号对应输出）的地址，之后 AX 和 3 进行与操作，因为这时只需考虑个位数是 0-3 的那个数来对应输出，之后转向形成地址。同样对于 1-35 的其他对应输出也是一样，将 AX 转换为最终要输出的数组的下标。在形成地址部分，需要先将 AX 值乘 4，因为输出数组是 DW 型，最终将数组首地址和偏移量对应过去进行输出即可。

4.针对输入转换的问题：

由于输入的时候，一般是采用输入字符串或者字符的形式，所以是以 ASCII 码形式输入到寄存器或者内存地址中的，所以需要先进行转换，一般是利用 AX 寄存器或者 BX 寄存器，然后根据 ASCII 码转换表，进行位操作，然后将输入的字符转换成字母或者数字，再进行操作，最

后输出的时候也会以 ASCII 码形式显示，所以也需要进行转换。