Lab 3 report

PB22111711 陈昕琪

实验目的与内容

编写一个LC-3机器语言的程序满足要求： 完成简化版本的strcmp()函数。

> strcmp ( )函数比较每个字符的ASCII值，直到找到非匹配值或找到NULL字符。

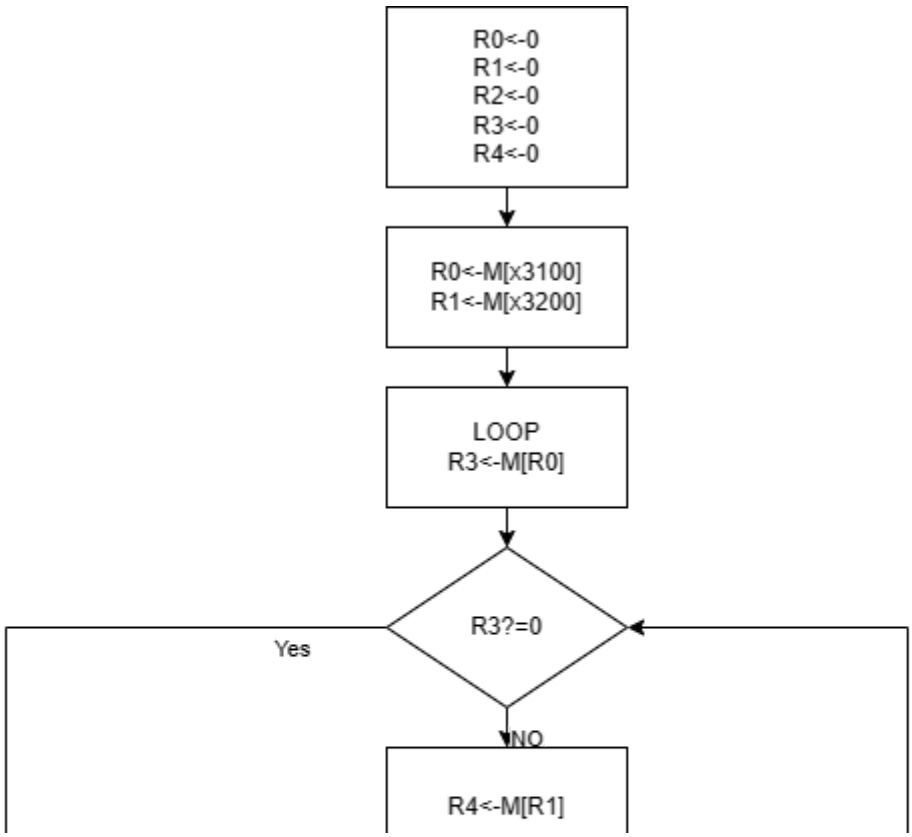
> 工作过程可以描述如下：

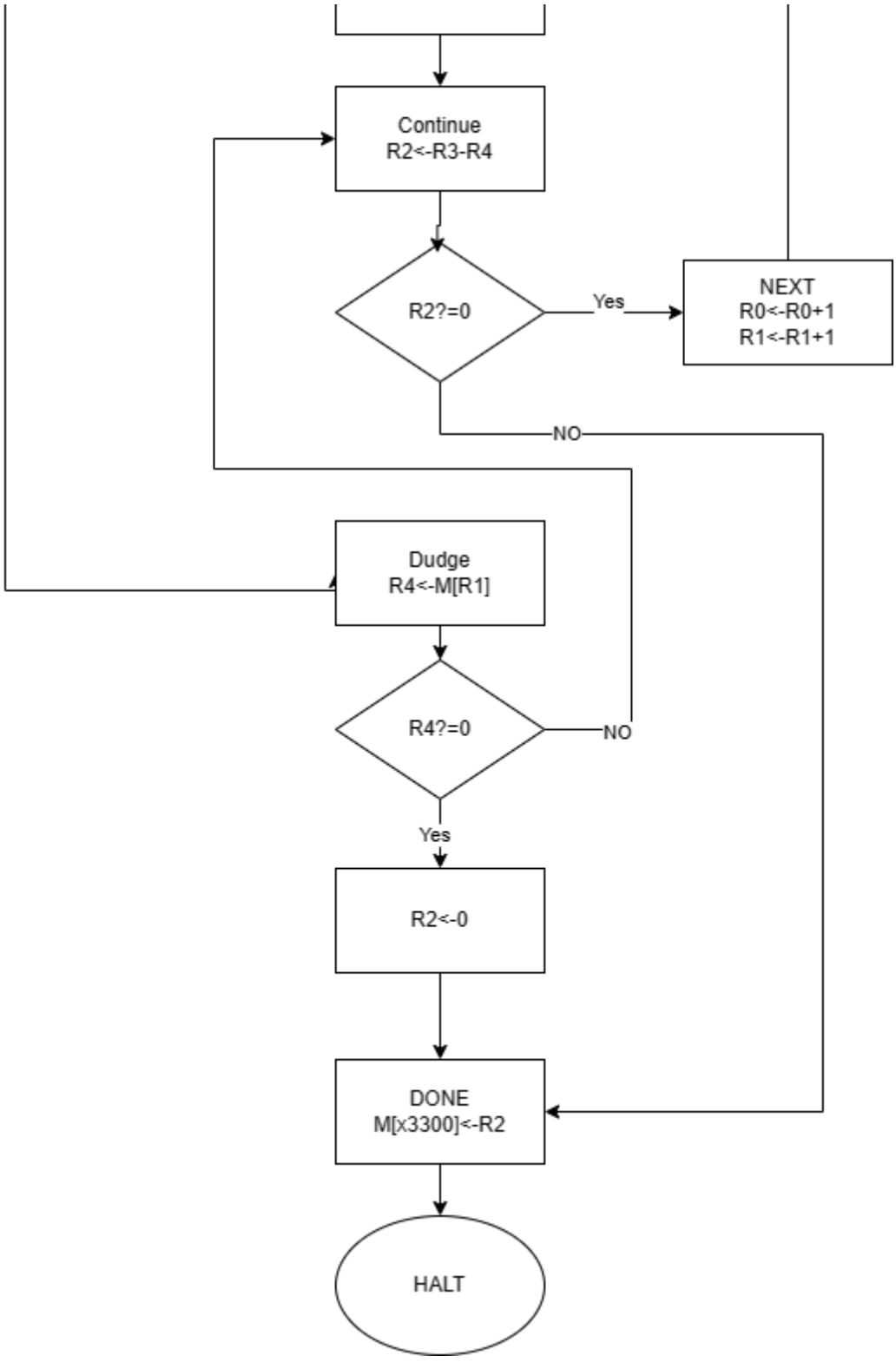
1. 首先比较两个字符串首字符的ASCII值。
2. 如果两个字符串中的第一个字符相等，那么这个函数将检查第二个字符，如果它们也相等，那么它将检查第三个字符，以此类推，直到检查到最后一个字符。
3. 如果发现非匹配字符，则返回第一个字符串中的字符的ASCII值减去第二个字符串中字符的ASCII值(返回的是带符号的值)

> S1和S2两个字符串的起始地址分别为x3100和x3200。 字符串中的每个字符都存储在连续的内存位置中，并且字符串以一个NULL字符结束。 假设S1和S2只包含来自a-z，A-Z，和一个NULL字符作为终结符。将strcmp()的返回值存储在x3300中。R0 - R7在开始时设置为零，程序在x3000开始。

逻辑设计

1. 程序主体思路： 将字符的首地址存入两个寄存器中，用循环操作依次比较两个字符串每一个字符的ASCII值的差。如果判断到两个字符串都结束(ASCII值都为0),则结果直接返回0。如果判断到两个字符的差值不为0，说明两个字符串不相等，并返回两个字符的差值。
2. 程序的流程图如下：





程序代码分析

1. 首先将各寄存器清空，将字符串首地址存入到相应的寄存器中

- R0用来存放第一个字符串的字符地址
- R1用来存放第二个字符串的字符地址
- R2用来存放两个字符串的字符差值，
- R3用来存放第一个字符串字符的ASCII值
- R4用来存放第二个字符串字符的ASCII值

```
.ORIG x3000;
AND R0, R0, #0;
AND R1, R1, #0;
AND R2, R2, #0;
AND R3, R3, #0;
AND R4, R4, #0;

LD R0, S1_ADDR;将字符串S1的地址存储在R0中
LD R1, S2_ADDR; 将字符串S2的地址存储在R1中
```

2. 然后，进行循环部分。

> 先将第一个字符串字符的值存入R3，然后需要判断R3是否为0，如果是0，则跳转到Judge语句判断R4是否为0。 如果R3不是0，则继续运算。将第二个字符串的字符值存入R4,获得R3减R4的值存入R2。

> **当R3不是0时，无需再判断R4的值是否是0，因为R4为0时，会返回R3-R4的值，R4不为0时，会继续判断R2是否为0。即无论R4是否是0，都进行下面的计算。无需再增加判断语句**

> 接下来判断R2，如果是0，跳转到NEXT语句，继续运算，如果不是0，则说明两个字符串出现字符不同的情况，则直接返回R2的值。

```
LOOP
LDR R3, R0, #0;
BRz Judge;判断R3的值是不是0，如果是0，则判断R4的值是不是0
LDR R4, R1, #0;当R3的值不是0时，可以继续运算，不需要考虑R4的值是否为0
Continue NOT R4, R4;
ADD R4, R4, #1;
ADD R2, R3, R4;R2=R3-R4
BRz NEXT;如果是0，则进行下一步
BRnp DONE;
```

3. Judge是在判断到R3为0时才运行的，先将第二个字符串对应的字符的值存进R4，然后需要判断R4的值是否为0。

> R4的值不为0，则返回Continue语句进行减法运算，得出的R2必然不为0，则会跳转到DONE语句。R4的值为0，则说明两个字符串相同，则将R2的值清零，并跳转到DONE语句。

```
Judge
LDR R4, R1, #0;
BRnp Continue;如果R4的值不是0则继续判断
AND R2, R2, #0;
BRnzp DONE;如果R3R4的值都为0则说明两个字符串相同
```

4. NEXT语句是判断字符串的对应字符相等，要比较下一个字符的时候才执行的语句。

> R0和R1是用于存放字符地址的值，所以只需要将R0和R1的值加一即可。

```
NEXT
ADD R0, R0, #1;
ADD R1, R1, #1;
BRnzp LOOP;
```

5. 最后，将相应的值储存在地址中，结束程序

```
DONE STI R2, RESULT; 结束循环，保存结果
HALT
```

测试结果与分析

测试结果图如下:

1. test1:

```
.ORIG x3100
S1 .STRINGZ "DsTAs"
.END

.ORIG x3200
S2 .STRINGZ "DstA"
.END
```

```
x3300          xFFE0          65504
```

2. test2:

```
.ORIG x3100
S1 .STRINGZ "DsTAs"
.END

.ORIG x3200
S2 .STRINGZ "DsTA"
.END
```

```
x3300          x0073          115
```

3. test3:

```
.ORIG x3100
S1 .STRINGZ "ABC"
.END

.ORIG x3200
S2 .STRINGZ "ABCD"
.END
```

```
x3300          xFFBC          65468
```

4. test4:

```
.ORIG x3100
S1 .STRINGZ "ABCD"
.END

.ORIG x3200
S2 .STRINGZ "ABC"
.END
```

```
x3300          x0044          68
```

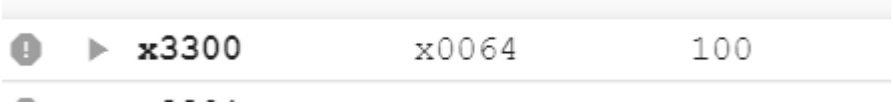
5. test5:

```
.ORIG x3100
S1 .STRINGZ "abc"
.END

.ORIG x3200
S2 .STRINGZ "abcd"
.END
```

```
x3300          xFF9C          65436
```

```
.ORIG x3100
S1 .STRINGZ "abcd"
.END

.ORIG x3200
S2 .STRINGZ "abc"
.END
```

6. test6:



由此可见程序正确。

遇到的问题及反思

本次实验过程中，遇到的问题较少，主要是各个情况判断的逻辑问题。

> 当判断到R3为0时，需要判断R4是否为0，而当R3不为0时，则不需要判断R4是否是0。

> 我没有单独将R0和R1(存放地址值的寄存器)自增的指令放进循环里，分开可以使程序更加清晰且可以优化条件跳转结构。

总结

1. 本次实验，通过编写lc3程序，深入了解了lc3语言，并巩固了课程中学习的lc3指令以及操作的运用。
2. 熟练掌握条件跳转语句的应用，并且可以根据程序初步优化结构，避免大量重复语句，使程序更简洁清晰。
3. 通过对程序的整体模块化划分和清晰的流程规划，可以做到减少debug的次数，提高程序的准确性。