lab3

实验目的

1. 进一步熟悉LC-3指令集.
2. 学会用汇编代码处理字符串.
3. 实现简单的`strcmp`函数.

实现方法

循环方式

分别使用`R0`和`R1`记录字符串0和字符串1下一个字符的地址, 每次使用`LDR`指令将两个字符载入`R3`和`R4`, 每轮循环结束让`R0`和`R1`自增. 代码如下:

```
        LD  R0, S0_ADDR  ; address of the next character of s0
        LD  R1, S1_ADDR  ; address of the next character of s1
LOOP    LDR R3, R0, #0   ; R3 <- char of s0
        LDR R4, R1, #0   ; R4 <- char of s1
                ...
        ADD R0, R0, #1   ; increment address
        ADD R1, R1, #1   ; increment address
        BR  LOOP         ; continue loop
```

比较字符

将`R4`取反加1, 与`R3`做ADD, 结果存入`R2`, 则`R2`即为两个字符的"距离". 代码如下:

```
ADD R4, R4, #1   ; R4 <- -R4
ADD R2, R3, R4   ; R2 <- char0 - char1
```

循环终止判断

1. 对于`NULL`字符, 仍按正常字符做比较.
2. 比较结果若不是0, 则跳转到`DONE`, 程序结束.
3. 否则, 再判断一下此时两个字符是否都为`NULL`, 是则程序结束, 否则程序继续.
   代码如下:

```
ADD R4, R4, #1   ; R4 <- -R4
ADD R2, R3, R4   ; R2 <- char0 - char1
BRnp DONE        ; if unmatching, done
```

```
ADD R3, R3, #0   ; set cc according to R3
BRz DONE         ; if char0 == char1 == NULL, done
```

结果存储

使用 `STI` 指令将 `R2` 的值存到 `x3300`.

```
DONE    STI R2, RESULT   ; write result
        HALT
```

整体代码

```
1         .ORIG   x3000
2
3         LD  R0, S0_ADDR  ; address of the next character of s0
4         LD  R1, S1_ADDR  ; address of the next character of s1
5   LOOP  LDR R3, R0, #0   ; R3 <- char of s0
6         LDR R4, R1, #0   ; R4 <- char of s1
7         NOT R4, R4
8         ADD R4, R4, #1   ; R4 <- -R4
9         ADD R2, R3, R4   ; R2 <- char0 - char1
10        BRnp DONE        ; if unmatching, done
11        ADD R3, R3, #0   ; set cc according to R3
12        BRz DONE         ; if char0 == char1 == NULL, done
13        ADD R0, R0, #1   ; increment address
14        ADD R1, R1, #1   ; increment address
15        BR  LOOP         ; continue loop
16
17  DONE  STI R2, RESULT   ; write result
18        HALT
19
20  S0_ADDR .FILL   x3100
21  S1_ADDR .FILL   x3200
22  RESULT  .FILL   x3300
23        .END
```

遇到的困难与解决方案

NULL字符的处理

1. 起初我打算在 `LDR` 指令结束后就使用 `BRz`, 但这样代码会很繁琐, 因为两次 `LDR` 都要一个 `BRz`, 并且还要分别对 `R2` 赋相应的值.
2. 经过分析, 我决定将 `NULL` 字符视为正常字符来处理, 因为无非两种情况:
   - 一个是 `NULL`, 一个不是. 此时比较结果将不为0, 会跳转到 `DONE`.
   - 两个都是 `NULL`, 此时比较结果为0, 那么在比较后再加一个 `BRz` 就可以了.
     由此, 得到了相对简洁的程序.

小bug

本次实验较简单, 代码较短, 因此没有出现太多问题. 唯一一处bug是我一开始手误将两个`LDR`的SR都打成`R0`了, 导致比较结果总是0. 经过单步调试, 我很快发现了问题并改正.

评测结果

使用如下几组测试数据:

```
IntroductionToComputingSystems:IntroductionToComputingSystem
icsICS:icsICs
QWERTYdfsaDFGHJKLMNBVCXZasfz:QWERTYdfsaBVCXZasfzDFGHJKLMN
helloWorld:(空字符串)
mYlittLeAirporT:mYlittLeAirp
(空字符串):(空字符串)
```

在自动评测机上的测试结果为:

汇编评测

6 / 6 个通过测试用例

- 平均指令数: 112
- 通过 IntroductionToComputingSystems:IntroductionToComputingSystem, 指令数: 328, 输出: 115
- 通过 icsICS:icsICs, 指令数: 64, 输出: -32
- 通过 QWERTYdfsaDFGHJKLMNBVCXZasfz:QWERTYdfsaBVCXZasfzDFGHJKLMN, 指令数: 119, 输出: 2
- 通过 helloWorld:, 指令数: 9, 输出: 104
- 通过 mYlittLeAirporT:mYlittLeAirp, 指令数: 141, 输出: 111
- 通过 :, 指令数: 11, 输出: 0

实验总结与收获

通过本次实验, 我:

1. 进一步熟悉了LC-3指令集, 以及汇编代码的编写.
2. 学会了如何用汇编代码处理字符串, 了解了计算机存储此类数据类型的方式.