Lab 3 Report

PB21081601 张芷苒

Purpose

使用 LC-3 汇编命令实现求字符串的最大重复子串。重复子串是一种由相同字符组成的子串。例如，“aabbbc”的重复子串为 “aa”、“bbb”和“c”。

- 字符串的长度 N 存储在内存位置 x3100
- 字符串的每个字符存储在以内存位置 x3101 开始的连续内存位置中
- 将最长重复子串的长度存储在内存位置 x3050

Principles

对于求最大重复子串问题，通常通过一次遍历来得到正确结果，即程序存储此时最大重复子串的长度以及当前重复子串的长度，若此时当前重复子串结束，则将其长度与最大长度比较，更新最大值。根据该思想可以得到伪代码：

```
Data: S, N
Result: length

left ← S[0];
right ← S[0];
index ← 1;
temp ← 1;
length ← 1;
N ← N − 1;
while N > 0 do
    right ← S[index];
    index ← index + 1;
    if left = right then
        temp ← temp + 1
    end
    else
        if temp > length then
            length ← temp
        end
        temp ← 1
    end
    left = right;
    N ← N − 1
end
```

Procedures

1. 初始化变量

从内存中读取字符串的长度 N，初始化字符串指针 。

```
INIT LDI R0, NUM ; N
    LD R1, DATA ; index
```

2. 处理 $N = 1$ 边界问题

将此时最大的重复子串长度初始化为 1，读取字符串中第一个字符，指针向后移动一位 。

```
LDR R2, R1, #0 ; right
    ADD R3, R2, #0 ; left
    ADD R1, R1, #1 ; index++
    AND R4, R4, #0 ;
    ADD R4, R4, #1 ; temp
    ADD R5, R4, #0 ; length
    ADD R0, R0, #-1 ; N--
```

3. 遍历字符串

遍历字符串，比较遍历到的字符与其前一个字符是否相同，若相同则该子串长度加一，若不同则跳转至 UPDATE，若循环结束则跳转至 UPDATE 进行最后一次更新。

```
WHILE BRz UPDATE
    LDR R2, R1, #0 ; right = S[index]
    ADD R1, R1, #1 ; index++
    ; if left = right
    NOT R6, R3
    ADD R6, R6, #1
    ADD R6, R2, R6
    BRnp UPDATE
    ADD R4, R4, #1 ; temp++
BACK ADD R3, R2, #0 ; left <- right
    ADD R0, R0, #-1 ; N--
    BR WHILE ; temp > length
```

4. 更新最大值

比较子串长度与当前最大重复子串长度，取较大值更新最大长度，并将子串长度重置为 1，判断此时是否结束遍历，若结束则跳转至 OUTPUT，若未结束则返回 WHILE。

```
UPDATE NOT R6, R4
    ADD R6, R6, #1
    ADD R6, R5, R6
    BRzp #1
    ADD R5, R4, #0 ; length <- temp
    AND R4, R4, #0
    ADD R4, R4, #1 ; temp <- 1
    ADD R0, R0, #0 ; N > 0
    BRz OUTPUT
    BR BACK
```

5. 存储结果

将最终结果存储到 x3050 。

```
OUTPUT STI R5, RESULT
```

Code

完整代码如下：

```
.ORIG x3000
INIT LDI R0, NUM ; N
    LD R1, DATA ; index
    LDR R2, R1, #0 ; right
    ADD R3, R2, #0 ; left
    ADD R1, R1, #1 ; index++
    AND R4, R4, #0 ;
    ADD R4, R4, #1 ; temp
    ADD R5, R4, #0 ; length
    ADD R0, R0, #-1 ; N--
WHILE BRz UPDATE
    LDR R2, R1, #0 ; right = S[index]
    ADD R1, R1, #1 ; index++
    ; if left = right
    NOT R6, R3
    ADD R6, R6, #1
    ADD R6, R2, R6
    BRnp UPDATE
    ADD R4, R4, #1 ; temp++
BACK ADD R3, R2, #0 ; left <- right
    ADD R0, R0, #-1 ; N--
    BR WHILE
    ; temp > length
UPDATE NOT R6, R4
    ADD R6, R6, #1
    ADD R6, R5, R6
```

```
    BRzp #1
    ADD R5, R4, #0 ; length <- temp
    AND R4, R4, #0
    ADD R4, R4, #1 ; temp <- 1
    ADD R0, R0, #0 ; N > 0
    BRz OUTPUT
    BR BACK

OUTPUT STI R5, RESULT

HALT

RESULT .FILL x3050
NUM .FILL x3100
DATA .FILL x3101
.END
```

Results

在测评网站上贴入代码，得到结果如下图：

汇编评测

3 / 3 个通过测试用例

- 平均指令数: 85.33333333333333
- 通过 6:aabbbc:3, 指令数: 92, 输出: 3
- 通过 5:ZZZZz:4, 指令数: 72, 输出: 4
- 通过 6:aabaaa:3, 指令数: 92, 输出: 3