作者：代兴　邮箱：503268771@qq.com　Vx 公众号：东方架构师

字符串比较器的功能

字符串比较器，输入 2 个字符串，比较字符串的大小，输出结果。
功能包括输入、输出、加法指令、函数指令、寻址指令、比较指令、跳转指令等。

字符串比较器的实现

编写代码：string_cmp.s

```
.data

byteArr_aa :
    .zero 100   # 字节数组。长度 100

byteArr_bb :
    .zero 100   # 字节数组。长度 100

str_input :     # 输入
    .string "%s  %s"

str_tip :       # 提示
    .string " >> Please input like :   hello  world \n"

str_bigger :    # 大于
    .string " output :  %s  >  %s \n\n"

str_smaller :   # 小于
    .string " output :  %s  <  %s \n\n"

str_equal :     # 等于
    .string " output :  %s  ==  %s \n\n"

.text
.global main

main :
    pushq %rbp
    movq %rsp, %rbp

    # 查看。
    movq $str_tip, %rdi
    callq printf
```

```
    # 输入。
    movq $str_input, %rdi
    movq $byteArr_aa, %rsi
    movq $byteArr_bb, %rdx
    callq scanf

# ------------------------------------------------
    movq $0, %r9            # 下标。依次遍历字节。

mark_loop :                 # 循环。
    movq $byteArr_aa, %rax  # 变量 aa 的地址
    addq %r9, %rax          # 找到当前的内存地址
    movb (%rax), %al        # 取一个字节
    movq $byteArr_bb, %rbx  # 变量 bb 的地址
    addq %r9, %rbx          # 找到当前的内存地址
    movb (%rbx), %bl        # 取一个字节

    cmpb $0, %al        # aa 到达末尾就退出循环
    je mark_out_loop
    cmpb $0, %bl        # bb 到达末尾就退出循环
    je mark_out_loop

    cmpb %bl, %al       # 比较单个字节的大小
    ja mark_bigger      # 大于
    jb mark_smaller     # 小于

    incq %r9            # 下标+1。遍历后续的字节
    jmp mark_loop       # 继续循环

# ------------------------------------------------
mark_out_loop :             # 退出循环。
    movq $byteArr_aa, %rax  # 变量 aa 的地址
    addq %r9, %rax          # 找到当前的内存地址
    movb (%rax), %al        # 取一个字节
    movq $byteArr_bb, %rbx  # 变量 bb 的地址
    addq %r9, %rbx          # 找到当前的内存地址
    movb (%rbx), %bl        # 取一个字节

    cmpb $0, %al        # aa 判断末尾
    jne mark_tail

    cmpb $0, %bl        # bb 判断末尾
    jne mark_tail

    jmp mark_equal      # aa、bb 同时到达末尾，说明相等

mark_tail :
    cmpb $0, %al        # aa 判断末尾
    je mark_smaller     # aa 更小
```

```
    cmpb $0, %bl        # bb 判断末尾
    je mark_bigger      # bb 更小

# ------------------------------------------------
mark_bigger :       # 大于
    movq $str_bigger, %rdi
    jmp mark_print

mark_smaller :      # 小于
    movq $str_smaller, %rdi
    jmp mark_print

mark_equal :        # 等于
    movq $str_equal, %rdi
    jmp mark_print

mark_print :        # 输出
    movq $byteArr_aa, %rsi
    movq $byteArr_bb, %rdx
    callq printf

    popq %rbp
    retq
```

编译代码：

```
gcc string_cmp.s -o string_cmp
```

运行代码：

```
[root@local zong]# ./string_cmp
 >> Please input like :   hello  world
apple orange
 output :  apple  <  orange

[root@local zong]# ./string_cmp
 >> Please input like :   hello  world
apple apple
 output :  apple  ==  apple

[root@local zong]# ./string_cmp
 >> Please input like :   hello  world
aabbcc aabbccddee
 output :  aabbcc  <  aabbccddee

[root@local zong]# ./string_cmp
 >> Please input like :   hello  world
aabbcc aaaaaa
 output :  aabbcc  >  aaaaaa
```

```
[root@local zong]# ./string_cmp
 >> Please input like :   hello  world
aaaa aa
 output :  aaaa  >  aa
```

分析结果：
输入 `apple orange` ，输出 `apple  <  orange` 。
输入 `apple apple` ，输出 `apple  ==  apple` 。
输入 `aabbcc aabbccddee` ，输出 `aabbcc  <  aabbccddee` 。
输入 `aabbcc aaaaaa` ，输出 `aabbcc  >  aaaaaa` 。
输入 `aaaa aa` ，输出 `aaaa  >  aa` 。

汇编代码	结果和分析

byteArr_aa :
    .zero 100   # 字节数组。长度 100
 	
byteArr_bb :
    .zero 100   # 字节数组。长度 100
	用变量承接 2 个输入字符串。

# 输入。 	
movq $str_input, %rdi
movq $byteArr_aa, %rsi
movq $byteArr_bb, %rdx
callq scanf
	使用 scanf 函数，输入变量。

movq $0, %r9 # 下标。依次遍历字节。 mark_loop : # 循环。	循环语句块。 设置下标，记录遍历的位置。

movq $byteArr_aa, %rax # 变量 aa 的地址
addq %r9, %rax # 找到当前的内存地址
movb (%rax), %al # 取一个字节
movq $byteArr_bb, %rbx # 变量 bb 的地址
addq %r9, %rbx # 找到当前的内存地址
movb (%rbx), %bl # 取一个字节
	从 2 个字符串各取 1 个字符。

cmpb $0, %al # aa 到达末尾就退出循环
je mark_out_loop
cmpb $0, %bl # bb 到达末尾就退出循环
je mark_out_loop
	判断循环的边界条件。
	字符串遍历到末尾就退出循环。

cmpb %bl, %al # 比较单个字节的大小 ja mark_bigger # 大于 jb mark_smaller # 小于	比较字符。 跳转指令使用无符号整数跳转。
incq %r9 # 下标+1。遍历后续的字节 jmp mark_loop # 继续循环	完成一次循环。 继续下一次循环。

cmpb $0, %al # aa 判断末尾
jne mark_tail
 	
cmpb $0, %bl # bb 判断末尾
jne mark_tail
	退出循环之后，判断字符串的末尾。
	如果末尾还有有效字符，就跳转到末尾语句块。
	否则，2 个字符串相同。

`    jmp mark_equal      # aa、bb 同时到达末尾，说明相等`	

`mark_tail :`
`    cmpb $0, %al        # aa 判断末尾`
`    je mark_smaller     # aa 更小`
 	
`    cmpb $0, %bl        # bb 判断末尾`
`    je mark_bigger      # bb 更小`
	如果某个字符串到达末尾了，说明更小。