团队任务 bf语言解释器文档（非logisim部分）

队名：组原组队

队员（花名）：NSC Lyt99 Sid shira13526 Subat

1 分工

NSC：完成bf解释器除输入输出以外部分的编写

Lyt99：完成bf解释器输入输出部分的编写并在logisim中增加对应逻辑

Sid、shira13526：完成测试样例的编写和测试

2 bf语言介绍

Brainfuck是一种极小化的计算机语言，它是由Urban Müller在1993年创建的，简称为BF。

它不像C++那么多元化，难度也一定的增加，因为它只有8种关键字。

BF语言可以和图灵机互相模拟

BF开了一个近乎无限长的数组，大小1byte，一个指针指向一个位置

2.1 bf基本操作

指针操作

> 指针右移一位指向下一个字节

< 指针左移一位指向上一个字节

字节操作

+ 当前指针指向的字节+1

- 当前指针指向的字节-1

读写操作

. 以char的形式输出当前指针指向的字节（48会输出'0'）

, 以ASCII的形式读入到当前指针指向的字节（读入’0'会存储48）

循环操作

[当当前指针指向的字节不是零时，会进行循环

] 当当前指针指向的字节非零时回到循环，否则退出

注：在当前实践例子中使用了$符号作为代码段的终止符

2.2 bf解释器

```
#  全局寄存器使用说明
#  $gp 程序指针
#  $fp 内存指针
#  $sp 栈指针
#  $s0-$s7 全局寄存器
#  # $s0 = '$' = 0x24 程序终止符（自定义）
#  # $s1 存放当前$gp指令内容
#  # $s2 存放当前$fp内存值
#  # $s3 存放当前$sp指向栈空间的值
#  # $s4 读入的用户输入
#  $t0-$t7 临时变量寄存器

#  系统调用说明
#  指令号在$v0中，分别为
#  0x32 停机
#  0x35 输入，值输入到$s4中
#  0x36 输出，值在$a0中


#  栈使用说明
#  $sp 栈指针
#  $s3 存放当前$sp指向栈空间的值
#  入栈操作
#  # addi    $sp, $sp, -4
#  # sw      $s3, 0($sp)
#  出栈操作
#  # lw      $s3, 0($sp)
#  # addi    $sp, $sp, 4

#  bf语言8个关键字对应ascii码 功能
#  > 0x3e 指针右移一位指向下一个字节
#  < 0x3c 指针左移一位指向上一个字节
#  + 0x2b 当前指针指向的字节+1
#  - 0x2d 当前指针指向的字节-1
#  . 0x2e 以char的形式输出当前指针指向的字节（48会输出'0'）
#  , 0x2c 以ASCII的形式读入到当前指针指向的字节（读入'0'会存储48）
#  [ 0x5b 当当前指针指向的字节不是零时，会进行循环
#  ] 0x5d 当当前指针指向的字节非零时回到循环，否则退出
```

```
.text
#  当前程序段功能：系统初始化
addi    $gp, $0, -4                 #  初始值为-4，work_loop开始会直接+4
add     $fp, $0, $0
addi    $s0, $0, 0x24               #  '$' = 0x24 程序终止符（自定义）

#  当前程序段功能：统计程序长度+8并存放在$fp中
#  当前程序段寄存器使用说明
#  # $s2 存放当前$fp内存值
count_loop:
lw      $s2, 0($fp)                 #  取当前bf命令
addi    $fp, $fp, 4                 #  指针自增
bne     $s0, $s2, count_loop        #  当前内容不等于终止符
addi    $sp, $fp, 60                #  栈顶，栈大小为15*4
add     $fp, $sp, $0                #  内存起始位置

#  当前程序段功能：程序执行
#  当前程序段寄存器使用说明
#  # $s1 存放当前$gp指令内容
#  # $s2 存放当前$fp内存值
#  # $s3 存放当前$sp指向栈空间的值
#  # $s4 存放当前[]对数
#  # $t0 临时存放bf关键字与$s1比较，随用随存
work_loop:
addi    $gp, $gp, 4
lw      $s1, 0($gp)                 #  取当前bf命令
beq     $s0, $s1, work_end          #  判断当前是否为终止符
#  > 0x3e 指针右移一位指向下一个字节
op_pointer_next:
addi    $t0, $0, 0x3e
bne     $s1, $t0, op_pointer_forward
addi    $fp, $fp, 4
j       work_loop
#  < 0x3c 指针左移一位指向上一个字节
op_pointer_forward:
addi    $t0, $0, 0x3c
bne     $s1, $t0, op_byte_add
addi    $fp, $fp, -4
j       work_loop
#  + 0x2b 当前指针指向的字节+1
```

```
op_byte_add:
addi    $t0, $0, 0x2b
bne     $s1, $t0, op_byte_sub
lw      $s2, 0($fp)
addi    $s2, $s2, 1
sw      $s2, 0($fp)
j       work_loop
#  - 0x2d 当前指针指向的字节-1
op_byte_sub:
addi    $t0, $0, 0x2d
bne     $s1, $t0, op_write
lw      $s2, 0($fp)
addi    $s2, $s2, -1
sw      $s2, 0($fp)
j       work_loop
#  . 0x2e 以char的形式输出当前指针指向的字节（48会输出'0'）
op_write:
addi    $t0, $0, 0x2e
bne     $s1, $t0, op_read
lw      $a0, 0($fp)
addi    $v0, $0, 0x36
syscall # 系统调用，输出到终端上
j       work_loop
#  , 0x2c 以ASCII的形式读入到当前指针指向的字节（读入'0'会存储48）
op_read:
addi    $t0, $0, 0x2c
bne     $s1, $t0, op_loop_in
addi    $v0, $0, 0x35
syscall
sw      $s4, 0($fp)
j       work_loop
#  [ 0x5b 当当前指针指向的字节不是零时，会进行循环
op_loop_in:
addi    $t0, $0, 0x5b
bne     $s1, $t0, op_loop_out
lw      $s2, 0($fp)
bne     $s2, $0, on_op_loop_in
add     $s4, $0, $0
#  不进行循环，找到循环结束符号']'
off_op_loop_in:
addi    $gp, $gp, 4                 #  指针自增
lw      $s1, 0($gp)                 #  取当前bf命令
addi    $t0, $0, 0x5d               #  在$t0存入']'
bne     $t0, $s1, judge_left        #  当前内容不等于']'
```

```
beq     $s4, $0, out_op_loop_in
addi    $s4, $s4, -1
judge_left:
addi    $t0, $0, 0x5b               #  在$t0存入'['
bne     $t0, $s1, off_op_loop_in    #  当前内容不等于'['
addi    $s4, $s4, 1
j       off_op_loop_in
out_op_loop_in:
j       work_loop
#  进行循环，当前$gp入栈
on_op_loop_in:
add     $s3, $gp, $0
addi    $sp, $sp, -4
sw      $s3, 0($sp)
j       work_loop
#  ] 0x5d 当当前指针指向的字节非零时回到循环，否则退出
op_loop_out:
addi    $t0, $0, 0x5d
bne     $s1, $t0, work_loop
lw      $s2, 0($fp)
bne     $s2, $0, on_op_loop_out
addi    $sp, $sp, 4
j       work_loop
on_op_loop_out:
lw      $gp, 0($sp)
j       work_loop

#  程序运行结束
work_end:
addi    $v0, $0, 50
syscall
```

2.3 测试样例

Helloworld.bf

```
++++++++[>++++++++>++++++++++>+++++<<<-]>.>++.+++++++..+++.>-.
------------.<++++++++.--------.+++.------.--------.>+.$
```

复读机.bf

```
+[,.]$
```

fibo.bf

```
store comma
>++++[-<++++++++++>]

read number
>, >++++++[-<--------->]>+++++++++[-<<<[->+>+<<]>>[-<<+>>]>]<<[-<+>],
>++++++[-<--------->]>+++++++++[-<<<[->+>+<<]>>[-<<+>>]>]<<[-<+>]
print first numbers (1)
<<+++++.-----.+++++.-----

initialize sequence
>-->+>+

start loop
<<
[-
  move / copy second
  >>[->+>+<<]
  sum first / second
  <[->>>+<<<]
  move second to first place
  >>[-<<+>>]
  move / copy sum
  >[-<<+>>>+<]
  print comma
  <<<<<.
  print number
  >>>>>>[>>>>++++++++++<<<<[->+>>+>-[<-]<[->>+<<<<[->>>+<<<]>]<<]>+[-
<+>]>>>[-]>[-<<<<+>>>>]<<<<]<[>++++++[<+++++++++>-]<-.[-]<]<<<<
]$
```