# 测试结果说明

一、PCODE指令集  
lit 0 a //把a放到栈顶

opr 0 a

//a=0 return

//a=1 取反

//a=2 加法

//a=3 减法

//a=4 乘法

//a=5 除法

//a=6 判断奇数odd

//a=8 判断等于

//a=9 判断不等

//a=10 小于

//a=11 大于等于

//a=12 大于

//a=13 小于等于

//a=14 输出

//a=15 换行

//a=16 输入

lod l a //把变量的值放到栈顶

sto l a //存入变量的值

cal l a //调用

Int 0 a //申请内存（SL DL RA+var）

jmp 0 a //无条件跳转

jpc 0 a //条件跳转

## 预测输入输出

输入：

1

输出：

1

1

1 2 3

输入：

2

输出：

2

2

3 4 5

## 三、控制台输入输出

<--代码生成开始-->

const a=3,b=4,c=5;

var x1x,s,a1,b1,c1;

procedure Circle;

begin

if a1=1 then

begin

s:=1;

write(s)

end

else

begin

a1:=a1-1;

call Circle

end

end;

2 INT 0 3 //分层输出PCODE（过程部分）

3 LOD 1 5

4 LIT 0 1

5 OPR 0 8

6 JPC 0 13

7 LIT 0 1

8 STO 1 4

9 LOD 1 4

10 OPR 0 14

11 OPR 0 15

12 JMP 0 18

13 LOD 1 5

14 LIT 0 1

15 OPR 0 3

16 STO 1 5

17 CAL 1 2

18 OPR 0 0

begin

a1:=a;

b1:=b;

c1:=c;

read(x1x);

write(x1x);

if x1x<2 then

begin

b1:=2;

c1:=3;

call Circle;

end

else

begin

s:=2;

write(s);

while x1x>=2 do

x1x:=x1x-1

end;

write(a1,b1,c1)

end.

19 INT 0 8 //分层输出PCODE（主程序部分）

20 LIT 0 3

21 STO 0 5

22 LIT 0 4

23 STO 0 6

24 LIT 0 5

25 STO 0 7

26 OPR 0 16

27 STO 0 3

28 LOD 0 3

29 OPR 0 14

30 OPR 0 15

31 LOD 0 3

32 LIT 0 2

33 OPR 0 10

34 JPC 0 41

35 LIT 0 2

36 STO 0 6

37 LIT 0 3

38 STO 0 7

39 CAL 0 2

40 JMP 0 55

41 LIT 0 2

42 STO 0 4

43 LOD 0 4

44 OPR 0 14

45 OPR 0 15

46 LOD 0 3

47 LIT 0 2

48 OPR 0 11

49 JPC 0 55

50 LOD 0 3

51 LIT 0 1

52 OPR 0 3

53 STO 0 3

54 JMP 0 46

55 LOD 0 5

56 OPR 0 14

57 LOD 0 6

58 OPR 0 14

59 LOD 0 7

60 OPR 0 14

61 OPR 0 15

62 OPR 0 0

<--代码生成结束--> //输出code[]中完整的PCODE代码

0 JMP 0 19 //跳到主程序

1 JMP 0 2 //跳到过程

2 INT 0 3

3 LOD 1 5

4 LIT 0 1

5 OPR 0 8 //a=1?

6 JPC 0 13 //a!=1跳到13

7 LIT 0 1

8 STO 1 4

9 LOD 1 4

10 OPR 0 14 //write s

11 OPR 0 15 //换行

12 JMP 0 18

13 LOD 1 5

14 LIT 0 1

15 OPR 0 3

16 STO 1 5

17 CAL 1 2 //call

18 OPR 0 0

19 INT 0 8 //主程序的开始申请了（SL DL RA +5var）=8个空间

20 LIT 0 3

21 STO 0 5

22 LIT 0 4

23 STO 0 6

24 LIT 0 5

25 STO 0 7

26 OPR 0 16 //read x1x to top

27 STO 0 3

28 LOD 0 3

29 OPR 0 14 //write x1x

30 OPR 0 15

31 LOD 0 3

32 LIT 0 2

33 OPR 0 10

34 JPC 0 41

35 LIT 0 2

36 STO 0 6

37 LIT 0 3

38 STO 0 7

39 CAL 0 2

40 JMP 0 55

41 LIT 0 2

42 STO 0 4

43 LOD 0 4

44 OPR 0 14

45 OPR 0 15

46 LOD 0 3

47 LIT 0 2

48 OPR 0 11

49 JPC 0 55

50 LOD 0 3

51 LIT 0 1

52 OPR 0 3

53 STO 0 3

54 JMP 0 46

55 LOD 0 5

56 OPR 0 14 //write

57 LOD 0 6

58 OPR 0 14

59 LOD 0 7

60 OPR 0 14

61 OPR 0 15

62 OPR 0 0 //结束

<--代码开始运行-->

input:1 //输入：1

1

1

1 2 3

<--代码运行结束-->//结果符合预期

//第二次运行

<--代码开始运行-->

input:2 //输入：2

2

2

3 4 5

<--代码运行结束-->//结果符合预期