试题分析

—camp

唐文斌

题意简述

- 生成一个CAM程序
  - 该程序可以从纸带上读取输入
  - 对指定表达式进行求值
  - 并将计算结果保存在纸带上
- 表达式含有1~3个运算符
- 可能含有加法、减法、乘法及小括号

CAM程序简介

- L <符号c>
  - 在当前单元格写入符号c (c = ?表示不改变)并左移
- R <符号c>
  - 在当前单元格写入符号c (c = ?表示不改变)并右移

- LOOP <符号表H>
- <循环体>
- END <符号表E>

CAM程序例子"a+1"

```
LOOP 0 1
        R ?
END ?
L ?                                  #移动到最后一位

LOOP 1
        L 0
END ?
L 1                                  #执行加1操作
```

数据分布

编号	表达式
1	a-1
2	a+b
3	a-b
4	a-b+c
5	a-(b+c)-d
6	(a-b)+(1+c)
7	a*b
8	a+b*c
9	a+b*(c-d)
10	(a+b)*(c+d)

得分情况

- 刘聪(山东) 34
- 陈瑜希(江苏) 30
- 周冬(安徽) 30
- 张煜承(江苏) 30
- 寿鹤鸣(安徽) 28
- 刘思壮(河北) 28

解法讨论

算法分析

- 基本思想——构造法
  - 构造基本的表达式运算
  - 将基本运算组合得到目标表达式

- 临时变量存储:
  - 内存(纸带) → 双端队列

后缀表达式

- a + b – c → a b + c –
- 优点
  - 存储空间为一个栈
  - 操作数都在栈顶
- 构造方法：遍历表达式树

算法分析(con't)

- 使用一侧作为栈存储空间(例如左侧)

	**(栈顶)	**	**(栈底)	a	b	c

- 模拟后缀表达式运算
  - 遇到操作数 → 压入栈中
  - 遇到操作符 → 取出两个操作符运算,结果入栈
- 对应到纸带机需要操作:
  - (1)写入常数1　　(2)拷贝变量值
  - (3)实现加、减、乘法　(4)保留答案

算法分析(con't)

- 由于操作是连续进行的，为了上下文无关，我们做如下约定：
  - 操作开始前(结束后)，读写头指向栈顶(最左侧)
  - 栈内的数值之间在纸带上用8分隔
- 任务：
  - 设计上述6种操作
  - 用到的一些设计技巧：
    - (1)if语句　(2)标记

技巧一：if语句

```
LOOP x                          #如果当前位置为x则执行
     DO SOMETHING
END                             #退出循环
```

技巧二：标记

- 用一个特别的字符代替原有字符，当操作完成时再改回
  - 目的：方便定位
  - 判断条件(控制循环) //见拷贝等
- 我们使用标记：
  - 用6/7标记已处理的0/1
  - 用4标记某一过程(循环)的结束

操作1:写入常数1

- 目标:
  - 向栈顶添加一个1

```
L ?    #移到栈顶左边一位
L 8    #添加分隔符
L 1    #写入1
R ?    #恢复栈顶指针
```

操作2：拷贝变量值

- 将某一个操作数从纸带中拷贝到栈顶
- 步骤1:在栈顶添加分隔符8
- 步骤2:定位到操作数的最低位
  - 根据栈顶元素个数(可由后缀表达式知)和待拷贝变量名，计算需要向右跳过多少个整数
  - 跳过一个整数的方法如下(跳过多个只需重复该代码)

```
L ?
R 8
L ?
```

```
LOOP 0 1
      R ?
END ?          #移动到一个分隔符8(或者9)
R ?            #移动到下一个数的第一位
```

操作2：拷贝变量值(con't)

- 步骤2:从最低位开始拷贝，使用"标记"
  - 用6/7标记已拷贝的0/1
  - 用4标记拷贝过程结束
  - (2.1)先移动到最后一个未被拷贝的bit

```
LOOP 0 1
     R ?
END ?        #移动到非0/1位置
L ?          #读写头返回
```

操作2：拷贝变量值(con't)

- (2.2)如果当前位是8，标记拷贝过程结束

```
LOOP 8
        R 4
        L ?
END                          #这是一个if语句
```

- (2.3/4)如果当前位是0/1，拷贝0/1到最前方

```
LOOP 0
        L 6                  #标记为6
        LOOP 0 1 8
                L ?
        END ?                #移动到最前方

        R 0                  #添加一个0
        L ?                  #恢复读写头位置
END
```

操作2：拷贝变量值(con't)

- 步骤3：在步骤2之外大循环，遇到4退出，此时读写头在4的位置
- 步骤4：还原标记

```
R ?                         #右移一位
LOOP 6 7
        LOOP 6              #6替换为0
                R 0
        END ?
        LOOP 7              #7替换为1
                R 1
        END ?
END ?
```

- 步骤5:读写头返回栈顶 (DONE)

操作3/4：加法(减法类似)

- 目标：在栈顶有加法操作数a，b，直接将右侧的b修改为答案a + b，并擦去a.
- 思路：从a的低位开始，逐位加到b中
- 在b中用”标记”,6/7表示处理过的0/1
- 步骤1：找到a的未操作的最低位

```
LOOP 0 1
        R ?
END ?
L ?
```

操作3：加法 (con't)

- 步骤2：将该位加到b中，由于a中这一位不会再用到，我们直接将其标记为8
  - (2.1) 如果该位为0，b不变
    - (2.1.1) 先定位到待操作的位

```
R 8
LOOP 8
        R ?
END ?                          #定位到第一个
LOOP 0 1
        R ?
END ?                          #定位到目标位
L ?
```

操作3：加法 (con't)

- (2.1.2) 对应位作相应标记

```
LOOP 1
       L 7
       R ?
END                          #if当前位为1, 改为7
LOOP 0 8
       L 6
END                          #如果是0, 则改为6
                             #8表示超过b的长度
LOOP 0 1 6 7
       L ?
END ?                        #回到b的首位之前
```

操作3：加法 (con't)

- (2.2) 如果该位为1，做对应加法
  - (2.2.1) 先定位到待操作的位
  - (2.2.2) 做加1操作，并处理进位

```
LOOP 1                  #case1 ： 如果被加位是1
       L 6              #加1之后变为0， 标记为6
       LOOP 1
              L 0
       END ?            #处理进位
       L 1              #最后的0改成1
       R ?              #将指针指向1(从而不与下面判断冲突)
END
LOOP 0 8                #case2 ： 如果被加位是0/8
       L 7              #加1之后变成1， 标记为7
END
LOOP 0 1 6 7            #返回b的首位之前
       L ?
END ?
```

操作3：加法 (con't)

- 步骤3：返回a的首位(沿着0,1,8)
- 步骤4：在步骤1,2,3外加一层大循环，处理a的每一位
- 步骤5：将a的无用位(均为8)擦除，并恢复指针

操作5：乘法

- 较为复杂，是前面几种操作设计的综合
- 思想：在栈顶添加一个数c,为原栈顶操作数a,b的乘积. 然后再将乘积c右移，删除a,b.
- 步骤1：将读写头移动到b的末位
- 步骤2：依次考虑b的每一位(使用标记等方式控制循环)
  - (2.1) 若b的最后一位为0，只需修改标记等
  - (2.2) 若b的最后一位为1，将a加到c对应位上

操作6：保留答案

- 当前答案在栈顶，我们仅需保留第一个数
- 步骤1：移动到第一个数后方

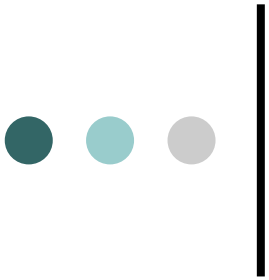
```
LOOP 0 1
       R ?
END ?                          #移动到第一个数后方
```

- 步骤2：擦除

```
LOOP 0 1 8
       R 9
END ?
```

主程序

- 对于读入表达式，构建后缀式
- 对于后缀式，依次产生相应的CAM程序



Thank You