**LAB 04**

**梁峻滔 PB19051175**

**1. 算法思想**

程序要求实现几个功能：

a) 打印输出每一行的石头

b）①提示Player 1/2 输入

②接收输入并回显

③检查输入的有效性并打印提示符

④修改石头数

c) 判断游戏是否结束和游戏结束时的赢家

每个功能都需要较多的指令和相对复杂的循环控制才能实现，因此将上述功能拆分成3个子程序来实现，再通过循环调用这3个子程序来完成游戏。

其中所有的打印输出都用TRAP x21实现，键盘输入都用TRAP x20实现，所有要输入/输出的字符都须用R0存放后才能使用TRAP x20/21指令。以下考虑各个功能如何实现。

**(1) PrintState子程序**

该子程序实现打印每一行的石头功能。首先需要打印“ROW A/B/C:”字符串，这三个字符串需要通过.STRINGZ伪指令设置存放在一片连续的内存空间，打印输出时用一个寄存器作为指针和R0读入字符后输出即可。问题是打印“ROW A:”之后随即需要打印石头‘o’，打印完A行的石头后再接着打印“ROW B:”，如何衔接起来？这里采用的方法是“ROW A:”“ROW B:”“ROW C:”存在连续的内存空间，这样，用R1作为指针，输出完“ROW A:”后R1继续+1，然后跳转至打印‘o’的子程序，打印完这一行的‘o’后再跳转回打印行提示符的子程序，这时R1已指向“ROW B:”字符串的首地址，就可以顺序打印出来，按同样的方法处理“ROW C:”，同时用R4作为一个计数器来判断是否三行都打印完即可。接下来是如何实现打印一定数量的‘o’。首先每一行现有的石头数需要三个内存位置来存放，这里是存放在x9001、x9002、x9003(实际上可以直接存放在对应的伪指令位置处，但由于敲代码时没有注意到，后面再修改会很麻烦就不做修改了)这三个地址分别存放在NumA、NumB、NumC中(用三条连续的.FILL伪指令指定)。用R0存‘o’的ASCII码，用另一个指针R2指向NumA/B/C，用R3读取A/B/C行现有的石头数后作为输出‘o’的计数器，R4继续作为行的计数器，就可以控制输出哪一行的以及多少个‘o’。检测到当前行石头数为0时，则不输出‘o’，另指针指向下一行。

**(2) Play子程序**

该子程序实现part(b)功能。

①提示Player1/2输入。使用.STRINGZ伪指令存放“Player1，choose a row…”“Player2，choose a row…”字符串，该字符串长度为42(包括‘\0’)，在调用者中预先设置好Player标识R6(为0时表示当前是Player1操作，为1时表示当前是Player2操作)，仍使用R1作为字符串指针，先存“Player1，…”的首地址，根据R6来判断是否需要令R1偏移(指向“Player2,…”)。用R0存R1指向的字符，再使用TRAP x21指令即可输出提示符。

②接收输入并回显。使用TRAP x20指令，随后跟一条TRAP x21指令即可。由于要输入“A1”“B2”“C6”这样一些大写字母和数字的组合，TRAP指令只能使用R0来接收一个字符，所以要

分两次输入，每次输入后都要先把R0接收的内容用其他寄存器存起来。

③检查输入的有效性并打印提示符。接收输入后，用输入的大写字母分别与‘A’‘C’比较，有效字母不能大于‘C’，不能小于‘A’，可以用ASCII码和NOT、ADD指令直接比较；用输入的数字跟指定行现有的石头数比较，输入的石头数不能小于1，不能大于该行现有的石头数。上述条件只要有一个不满足，该输入就是无效的。输入无效时跳转输出“Invalid…”字符串后重新跳转回开头，然后提示输入、接收输入。

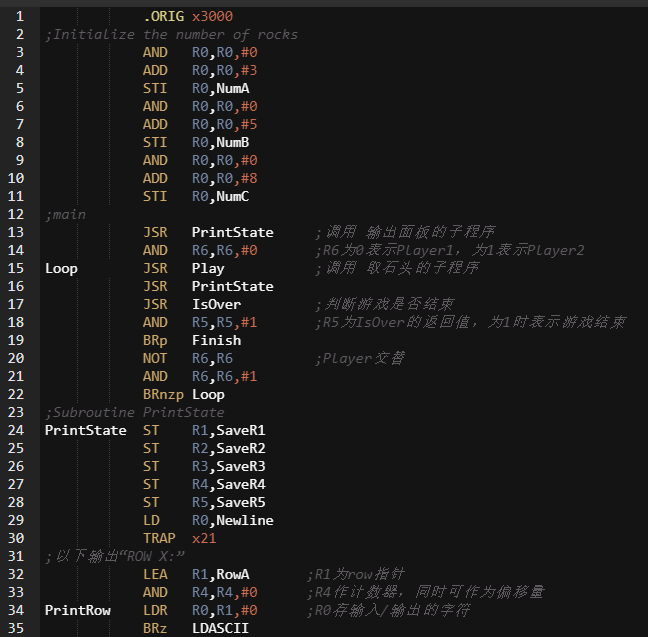
④修改石头数。预先通过间接寻址读入A行存放石头数的位置的地址，通过读入的大写字母的ASCII码与‘A’的ASCII码相减来得到偏移量，指针偏移后读入该行的石头数，对读入的要取走的石头数进行ASCII码到数值的转换，相减，判断有效时直接将相减的结果存回该行存放石头数的位置。

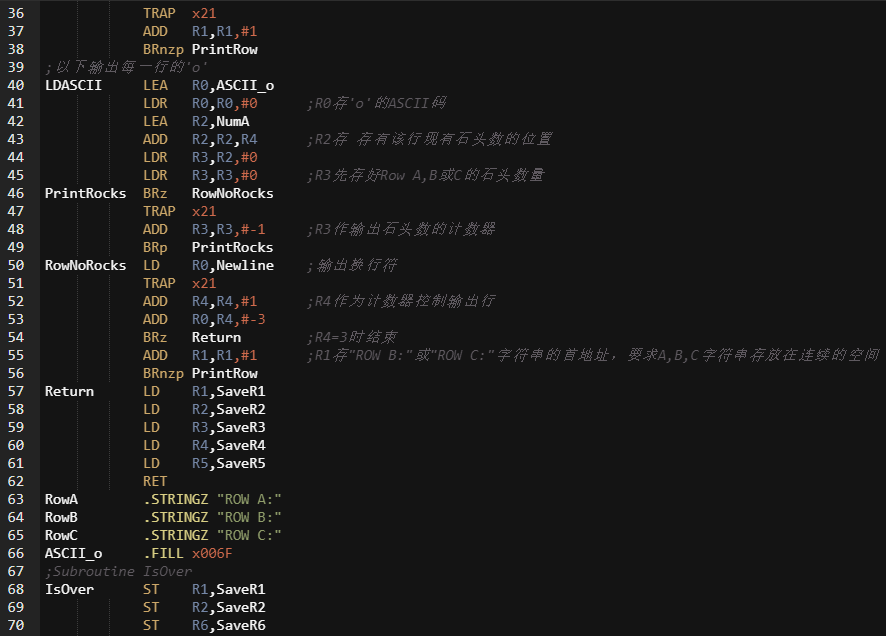
**(3) IsOver子程序**

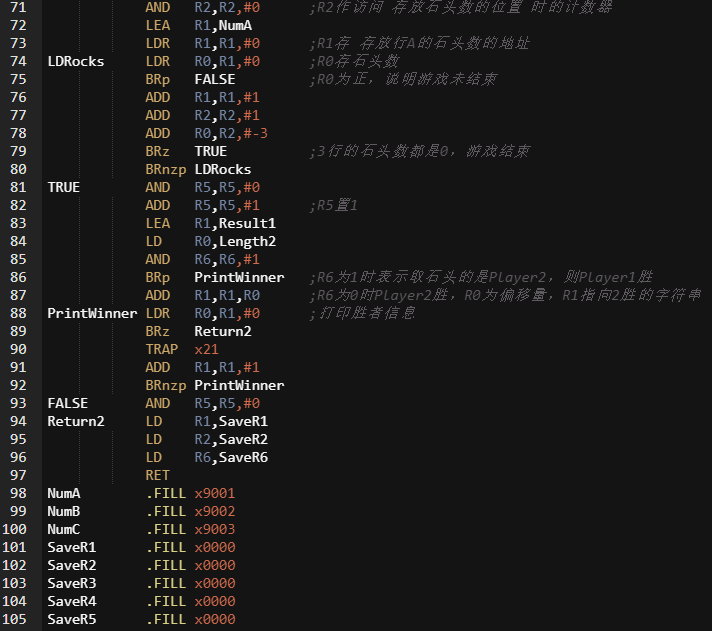
该子程序判断游戏是否结束和游戏结束时的赢家。在调用者中，每次调用完Play和PrintState后就调用此程序，R6继续作为输入，标识是哪一个Player执行了Play程序。游戏是否结束通过判断是否每一行的石头数都为0来判断，当游戏结束时R5置1，然后输出赢家的提示符信息；游戏未结束时R5置0。当R5返回值为0时，在调用者中交换Player(修改R6)，然后继续调用Play程序。

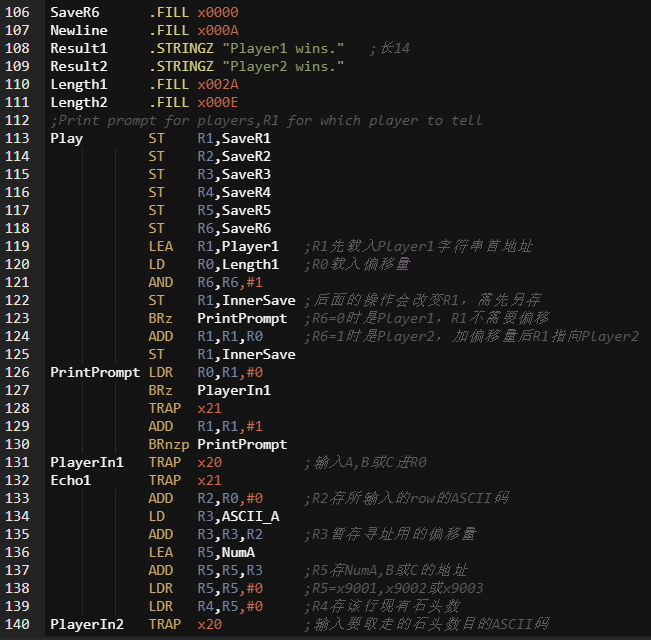
如此，三个主要的子程序和主程序都搭好了。

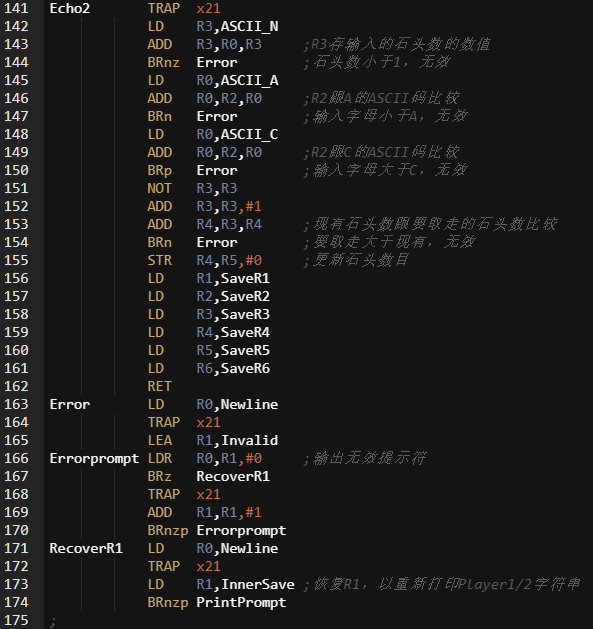
**2. 代码实现**

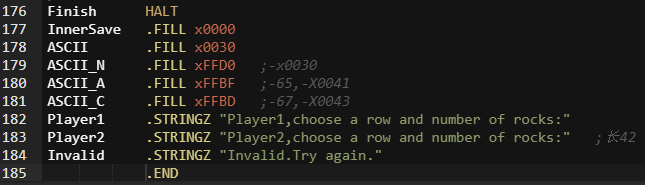
****

****

****

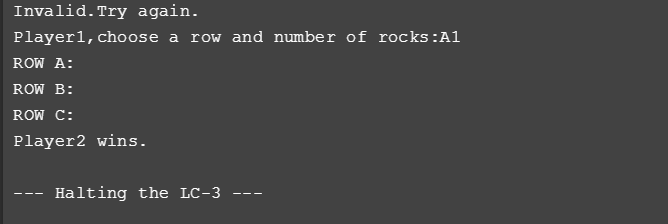
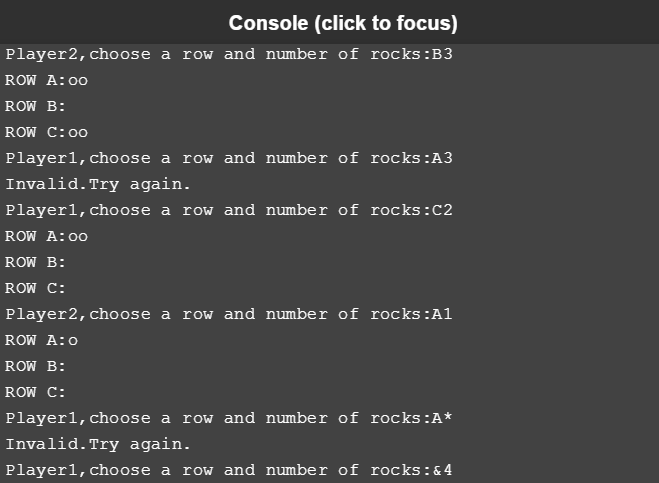
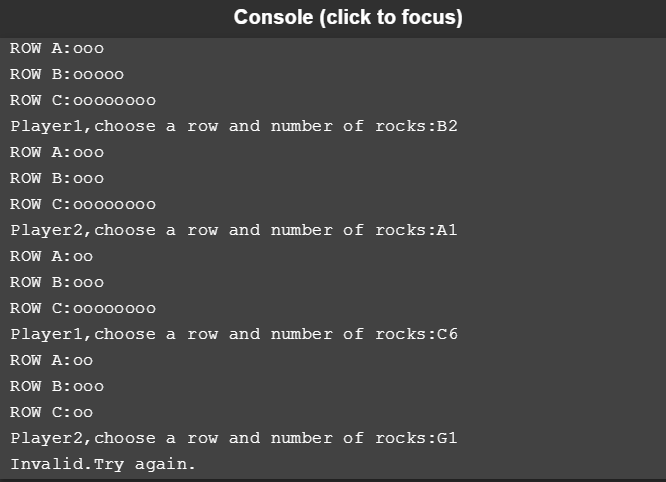
****

****

****

**3. 测试**

按照所给Example输入，得到的结果如下，与Example中的过程和结果都一致。

****