python 实现的LC3汇编器和模拟器

邱子悦 PB18000028 2019.12.17

核心思想

- 筛掉无用的空格、换行符和注释（有许多关于引号和分号的特殊情况要考虑）
- 取出每行第一个词，建表（注意标识符的语法要求）
- 判断所有语句的语法，逐行报错
- 如果没有语法错误，逐行翻译并二进制写入.obj文件

具体实现

读.asm文件

```
# read the file
# list1 should be used later
f1 = open("try.asm","r",encoding='utf-8')
list1=f1.readlines()
f1.close()
```

清除空格、换行符和注释

```
# strip the notes and '\n' and the ' '
# list3 should be used later
# use count
count = 0
list2=[]
list3=[]
for line1 in list1:
    if(line1.isspace()):
        continue
    line1=line1.replace('\t'," ").replace('\n',"")
    line1=line1.strip(' ')
```

【亮点1】考虑".STRINGZ"语句后字符串的特殊性，不能直接用 “;” split

利用正则表达式提取需要的部分

```
re.findall('.*.STRINGZ.*\"',line1)
```

这句的含义是：如果存在，从头取到最后一个双引号

之后进一步确定是否只有两个合适的引号

我使用的测试样例是：

```
.STRINGZ "euwhd;iw\"ss" "sos" ;wkdw"dfo
```

详细代码（略）

```
```

```
    # there can be ';' in the .STRINGZ, so use re to find it
    # test example: "euwhd;iw\"ss" "sos" ;wkdw"dfo
    if(len(re.findall('.*.STRINGZ.*\"',line1))!=0):
        line1=re.findall('.*.STRINGZ.*\"',line1)[0]
        if(';' not in line1): # still need to find some error
            flag_yinhao=0
            for nu in range(len(line1)):
                if(flag_yinhao == 2):
                    count=count+1
                    print("Unrecognized opcode or syntax error after .STRINGZ")
                    break
                if(line1[nu]=='\"' and flag_yinhao == 0): # first yinhao
                    flag_yinhao=1
                elif(line1[nu]=='\"' and line1[nu-1]!='\\' and flag_yinhao ==
1):
                    flag_yinhao=2
            list3.append(line1)
        else:
            flag_yinhao=0
            for nu in range(len(line1)):
                if(flag_yinhao == 2):
                    count=count+1
                    print("Unrecognized opcode or syntax error after .STRINGZ")
                    break
                if(line1[nu]=='\"' and flag_yinhao == 0): # first yinhao
                    flag_yinhao=1
                elif(line1[nu]==';' and flag_yinhao == 1): # ignore
                    continue
                elif(line1[nu]=='\"' and line1[nu-1]!='\\' and flag_yinhao ==
1):
                    flag_yinhao=2
            line1=line1[0:nu+1]
            list3.append(line1)
    else:
        list2=line1.split(';')
        if(len(list2[0])!=0):
            list3.append(list2[0])
```

建立了Total_Words的列表，建立标识符对应行数的dictionary

【亮点2】考虑了标识符的构成语法：下划线或字母开头，所有字符均为下划线、字母或数字

```
# find out the notations and build up a dictionary for them
Instruction=["ADD","AND","BR","BRn","BRz","BRp",
             "BRnz","BRzn","BRnp","BRpn","BRpz",
             "BRzp","BRnzp","BRzpn","BRznp","BRnpz",
             "BRpzn","BRpnz","JMP","JSR","JSRR",
             "LD","LDI","LDR","LEA","NOT","RET",
             "RTI","ST","STI","STR","TRAP","HALT",
             "GETC","OUT","PUTS","IN","PUTSP"]
Pseudo_ops=[".ORIG",".FILL",".BLKW",".STRINGZ",".END"]
Total_Words=Instruction + Pseudo_ops

nota_dict={}
for i in range(len(list4)):
    for word in Total_Words:
        if(word==list4[i][0]):
```

```
            break
    else:
        # start with '_' or alpha
        # all alpha/_/number
        stri=list4[i][0]
        if(stri.replace('_','').isalnum() and (stri[0].isalpha() or
stri[0]=='_')):
            nota_dict[list4[i][0]]=i
        else:
            count=count+1
            print("Invalid label",stri)
```

【亮点3】利用int存储16位数字 (可利用位运算)，之后再利用bytearray写入.obj文件

两个小函数:

```
f2=open(r'try.obj',mode='wb')
def IMM(num):
    if(num[0]=='#'):
        return eval(num[1:])
    elif(num[0]=='x'):
        return int(num[1:],16)
    elif(num.isdigit()):
        return eval(num)
    else:
        return num

def write_array(array):
    byte_arr=[int(array/256),array%256]
    f2.write(bytearray(byte_arr))
```

检查语法错误并报错

部分代码 (AND 和 ADD) :

```
if(list2[0]=="ADD" or list2[0]=="AND"):
        if(len(list2)>=4):
            if(len(list2)>4):
                count=count+1
                print("line",i,"Unrecognized opcode or syntax error")
            if(list2[1][0]!='R'):
                count=count+1
                print("line",i,"Expected register operand, but
found",list2[1],"instead")
            elif(list2[1][1:] not in ['0','1','2','3','4','5','6','7']):
                count=count+1
                print("line",i,"Expected register operand, but
found",list2[1],"instead")
            if(list2[2][0]!='R'):
                count=count+1
                print("line",i,"Expected register operand, but
found",list2[2],"instead")
            elif(list2[2][1:] not in ['0','1','2','3','4','5','6','7']):
                count=count+1
                print("line",i,"Expected register operand, but
found",list2[2],"instead")
            # register or IMM5
```

```
            if(list2[3][0]=='R' and list2[3][1:] not in
['0','1','2','3','4','5','6','7']):
                count=count+1
                print("line",i,"Expected register or immediate value, but
found",list2[3],"instead")
            elif(list2[3][0]!='R' and IMM(list2[3]) not in range(-32,32)):
                count=count+1
                print("line",i,"Expected register or 5 bit immediate value, but
found",list2[3],"instead")
        else:
            count=count+1
            print("line",i,"Lack of operand")
```

【亮点4】基本复现了所有情况的报错，对于部分报错改为了更自然的方式

部分测试样例：

```
.ORIG
AND R9,R1
1A RET
LD R1,R1,R1
.END 18993
```

报错：

```
Invalid label 1A
line 0 Expected 16 bit value
line 1 Lack of operand
line 3 Unrecognized opcode or syntax error
line 3 Expected label or 9 bit immediate value, but found R1 instead
5 error(s)
```

LC3Edit (某行行尾的错误，会报下一行的错)

```
Starting Pass 1...
Line 2: Expected 16 bit value, but found 'AND' instead
Line 2: Register 9 does not exist
Line 3: Expected register operand, but found '1A' instead
Line 5: Unrecognized opcode or syntax error at or before '.END'
Pass 1 - 4 error[s]
```

【亮点5】利用位运算，更高效地生成了数据

值得一提的是负数，比如利用&0x1ff就能实现取出后9位，再加在array上。

一开始我还忘了取位移量的 "((nota_dict[list2[1]]-i-1)&0x1ff)" 也要加上取后9位。

部分代码 (BR)：

```
    elif(list2[0] in BRlist):
        array=0
        if(len(list2[0])==2):
            array=array+int('111',2)
        if('n' in list2[0]):
            array=array+int('100',2)
        if('z' in list2[0]):
            array=array+int('10',2)
        if('p' in list2[0]):
```

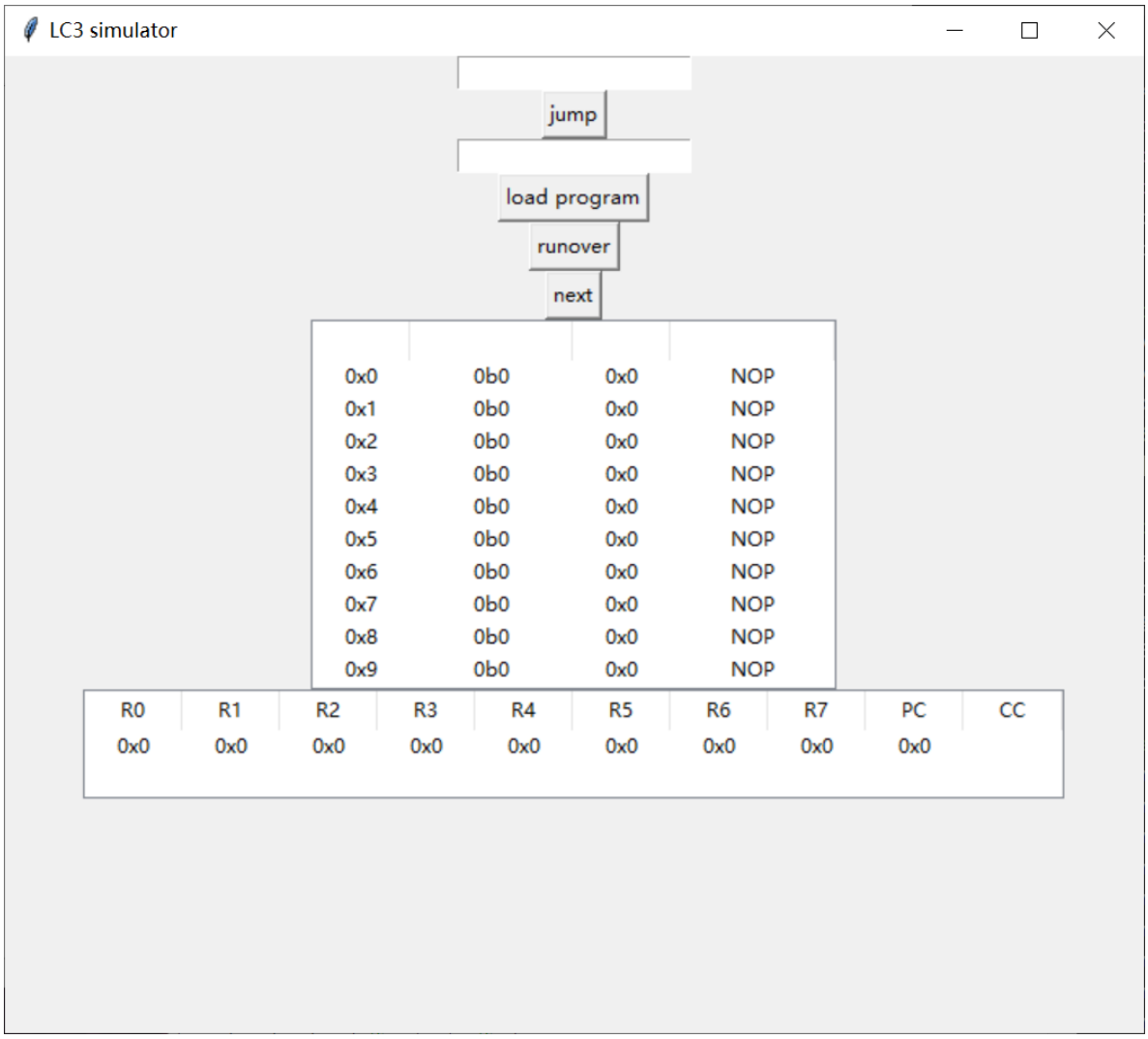
```
            array=array+int('1',2)
        array=array<<9
        if(list2[1][0]=='#' or list2[1][0]=='x' or list2[1].isdigit()):
            array=array+(IMM(list2[1])&0x1ff)
        else:
            array=array+((nota_dict[list2[1]]-i-1)&0x1ff)
```

【亮点6】"麻烦的" .STRINGZ 字符串的显式与隐式转化

```
    elif(list2[0]=='.STRINGZ'):
        list2[1]=list2[1][1:-1]
        for k in range(len(list2[1])):
            if(list2[1][k:k+2]=="\\n"):
                if(k==0):
                    list2[1]='\n'+list2[1][2:]
                else:
                    list2[1]=list2[1][0:k]+'\n'+list2[1][(k+2):]
            elif(list2[1][k:k+2]=="\\r"):
                if(k==0):
                    list2[1]='\r'+list2[1][2:]
                else:
                    list2[1]=list2[1][0:k]+'\r'+list2[1][(k+2):]
            elif(list2[1][k:k+2]=="\\t"):
                if(k==0):
                    list2[1]='\t'+list2[1][2:]
                else:
                    list2[1]=list2[1][0:k]+'\t'+list2[1][(k+2):]
            elif(list2[1][k:k+2]=="\\\'"):
                if(k==0):
                    list2[1]='\''+list2[1][2:]
                else:
                    list2[1]=list2[1][0:k]+'\''+list2[1][(k+2):]
            elif(list2[1][k:k+2]=='\\\"'):
                if(k==0):
                    list2[1]='\"'+list2[1][2:]
                else:
                    list2[1]=list2[1][0:k]+'\"'+list2[1][(k+2):]
            elif(list2[1][k:k+2]=="\\\\"):
                if(k==0):
                    list2[1]="\\"+list2[1][2:]
                else:
                    list2[1]=list2[1][0:k]+"\\"+list2[1][(k+2):]
            elif(list2[1][k:k+2]=="\\0"):
                if(k==0):
                    list2[1]="\0"+list2[1][2:]
                else:
                    list2[1]=list2[1][0:k]+"\0"+list2[1][(k+2):]
        for ch in list2[1]:
            f2.write(bytearray([0]))
            f2.write(bytearray(ch,'utf-8'))
        array=0
        write_array(array)
```

测试时间!

邱子悦 PB18000028 2019.12.21



实现功能

- 可以跳转任意位置
- 导入程序或数据
- 单步
- 运行到HALT停止
- 显示内存的情况和R0~7，PC，CC

具体实现

初始化：

```
# initial
memory = np.zeros((65536),dtype=np.uint16)
memory_asm = ["NOP"]*65536
R = np.zeros((8),dtype=np.int16)
PC = 0
CC=[0]*3

# initial draw
root = Tk()
root.title("LC3 simulator")
root.geometry('700x600')
```

内存区域图形化：

```
# treeview
tree = ttk.Treeview(root,show="headings",height=10)
tree["columns"]=("address in hex","value in bin","value in hex","asm")
tree.column("address in hex",width=60, anchor='center')
tree.column("value in bin",width=100, anchor='center')
tree.column("value in hex",width=60, anchor='center')
tree.column("asm",width=100, anchor='center')
```

转换成asm（部分代码）：

```
def ExchangeToAsm(unsh_address, unsh_get):
    # global
    global memory
    global memory_asm
    # unsh_get is a unsigned 16 bit int
    LABEL = (unsh_get & 0xf000)>>12
    # NOP
    if(unsh_get & 0xfe00 == 0):
        sys.exit(0)
    else:
        memory_asm[unsh_address] = ""
    # ADD AND
    if(LABEL==int('0001',2) or LABEL==int('0101',2)):
        if(LABEL==int('0001',2)):
            memory_asm[unsh_address]+="ADD "
        else:
            memory_asm[unsh_address]+="AND "
        DR = (unsh_get & 0xe00)>>9
        memory_asm[unsh_address]+="R"+"{:}, ".format(DR)
        SR1 = (unsh_get & 0x1c0)>>6
        memory_asm[unsh_address]+="R"+"{:}, ".format(SR1)
        if((unsh_get & 0x20)>>5):
            IMM5 = exchangeNeg(5, unsh_get)
            memory_asm[unsh_address]+="#{:}".format(IMM5)
        else:
            SR2 = unsh_get & 0x7
            memory_asm[unsh_address]+="R"+"{:}".format(SR2)
        #print(memory_asm[unsh_address])
```

刷新屏幕

```
def newit():
    global tree
    global treeshow
    global R
    global PC
    global CC
    x=tree.get_children()
    for item in x:
        tree.delete(item)
    for i in range(PC,PC+10):
        tree.insert('',i,values=
(hex(i),bin(memory[i]),hex(memory[i]),memory_asm[i]))
        """
```

```
        if(i == PC):
            tree.item(tree.get_children()[0],tags='PC')
            tree.tag_configure('PC',background='yellow')"""
    tree.pack()
    y=treeshow.get_children()
    for item in y:
        treeshow.delete(item)
    treeshow.insert('',0,values=
(hex(R[0]),hex(R[1]),hex(R[2]),hex(R[3]),hex(R[4]),hex(R[5]),hex(R[6]),hex(R[7])
,hex(PC),'N'*CC[0]+'Z'*CC[1]+'P'*CC[2]))
    treeshow.pack()
```

单步

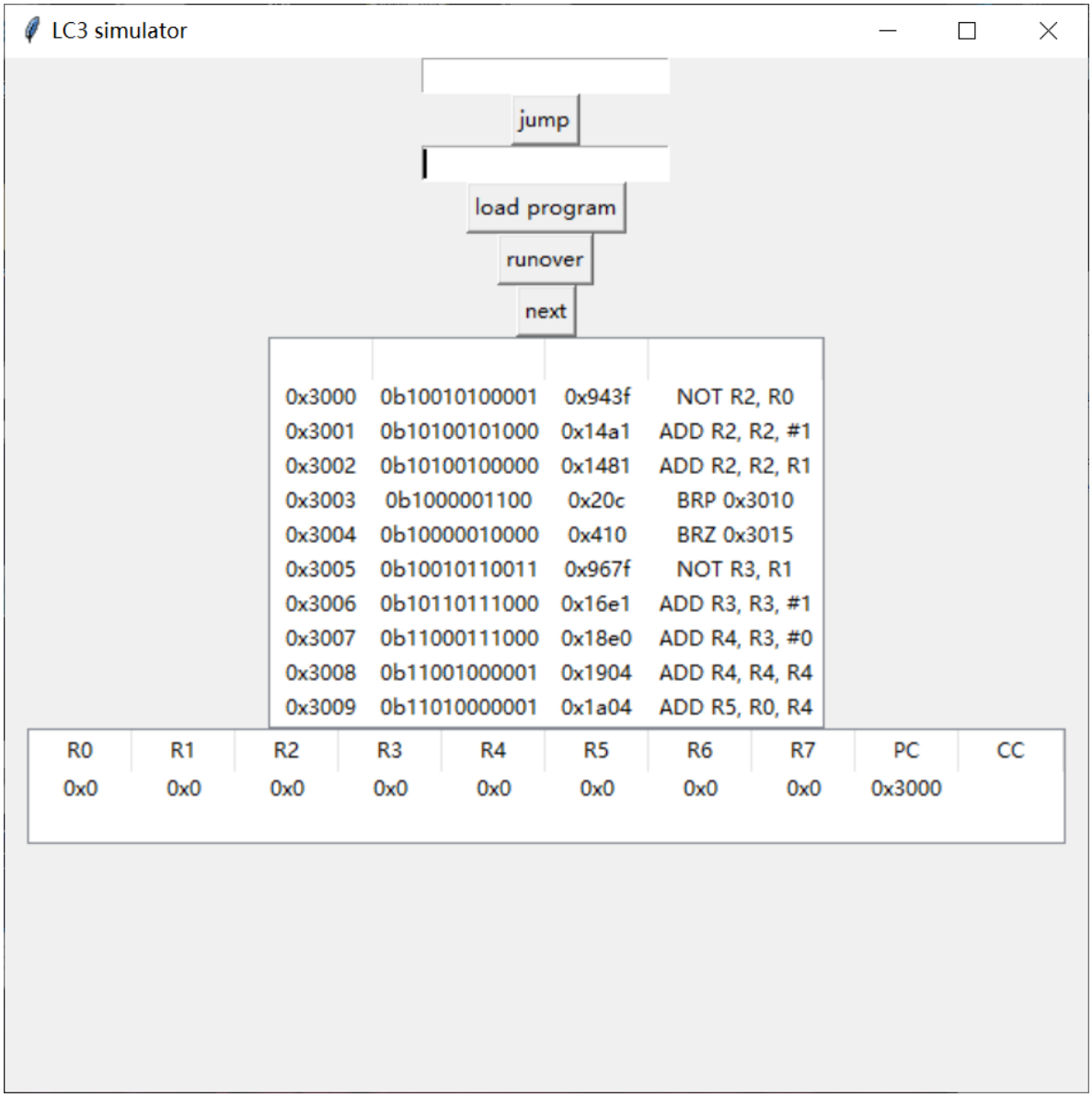
```
def next():
    # global
    global memory
    global PC
    global CC
    global R
    # PC
    PC+=1
    # unsh_get is a unsigned 16 bit int
    unsh_get=memory[PC-1]
    LABEL = (unsh_get & 0xf000)>>12
    # ADD AND
    if(LABEL==int('0001',2) or LABEL==int('0101',2)):
        DR = (unsh_get & 0xe00)>>9
        SR1 = (unsh_get & 0x1c0)>>6
        if(LABEL==int('0001',2)):
            if((unsh_get & 0x20)>>5):
                IMM5 = exchangeNeg(5, unsh_get)
                R[DR]=R[SR1]+IMM5
            else:
                SR2 = unsh_get & 0x7
                R[DR]=R[SR1]+R[SR2]
        else:
            if((unsh_get & 0x20)>>5):
                IMM5 = exchangeNeg(5, unsh_get)
                R[DR]=R[SR1] & IMM5
            else:
                SR2 = unsh_get & 0x7
                R[DR]=R[SR1] & R[SR2]
        SetCC(R[DR])
```

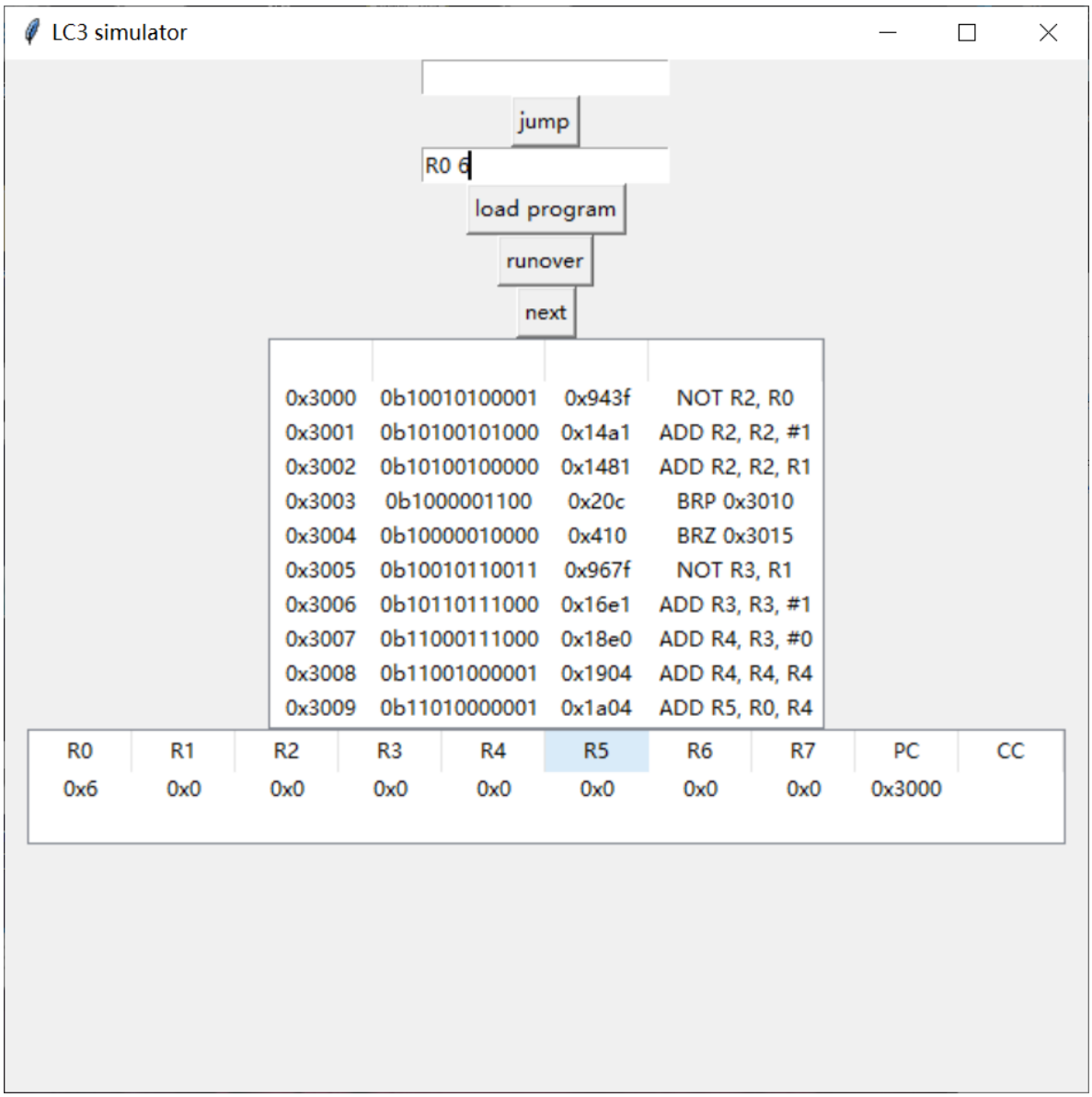
给寄存器赋值

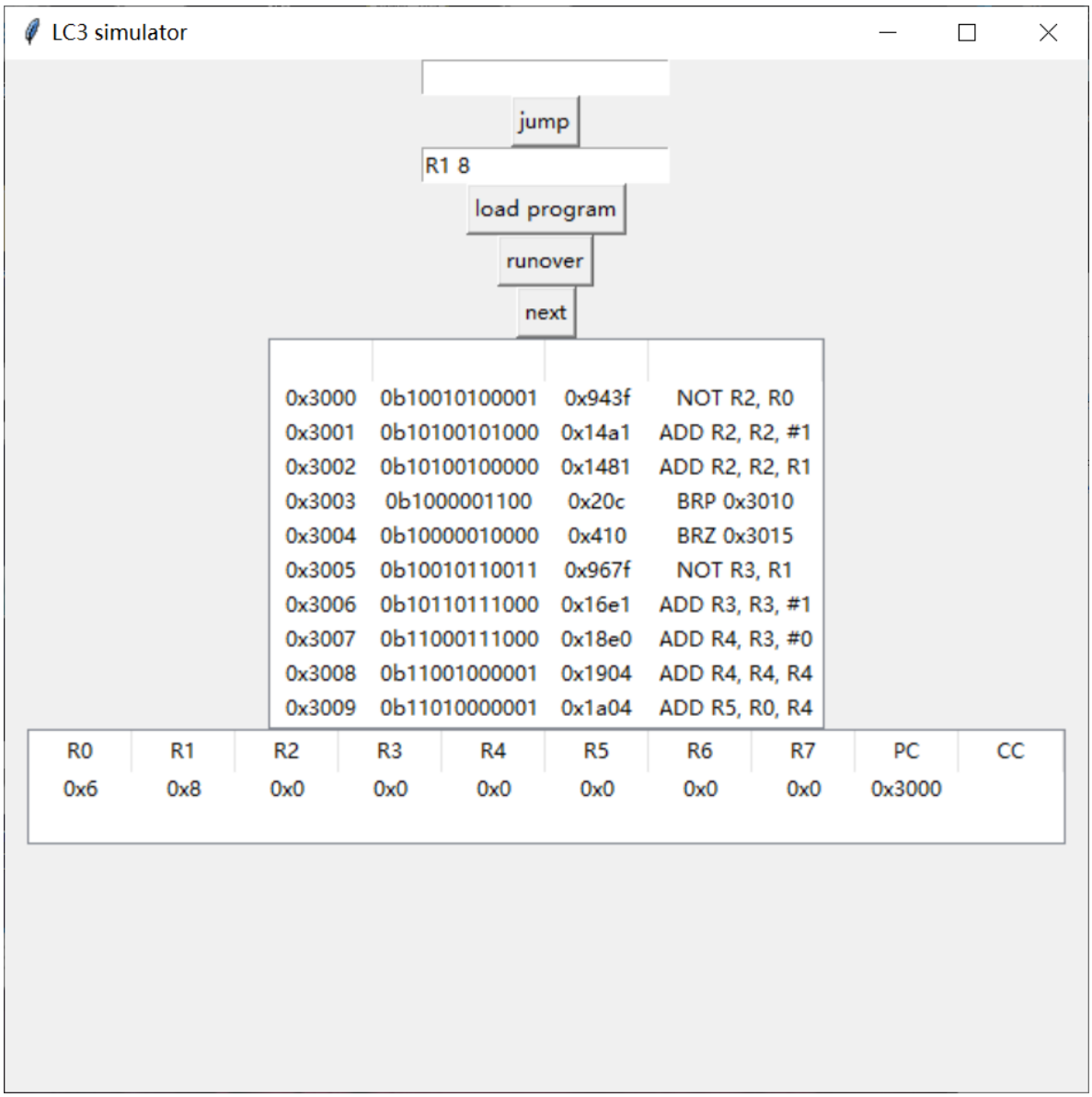
```
def Show(event):
    global treeshow
    global ent
    global R
    l = ent.get().split()
    R[eval(l[0][1:])]=int(l[1],16)
    newit()

def SetShow(event):
```

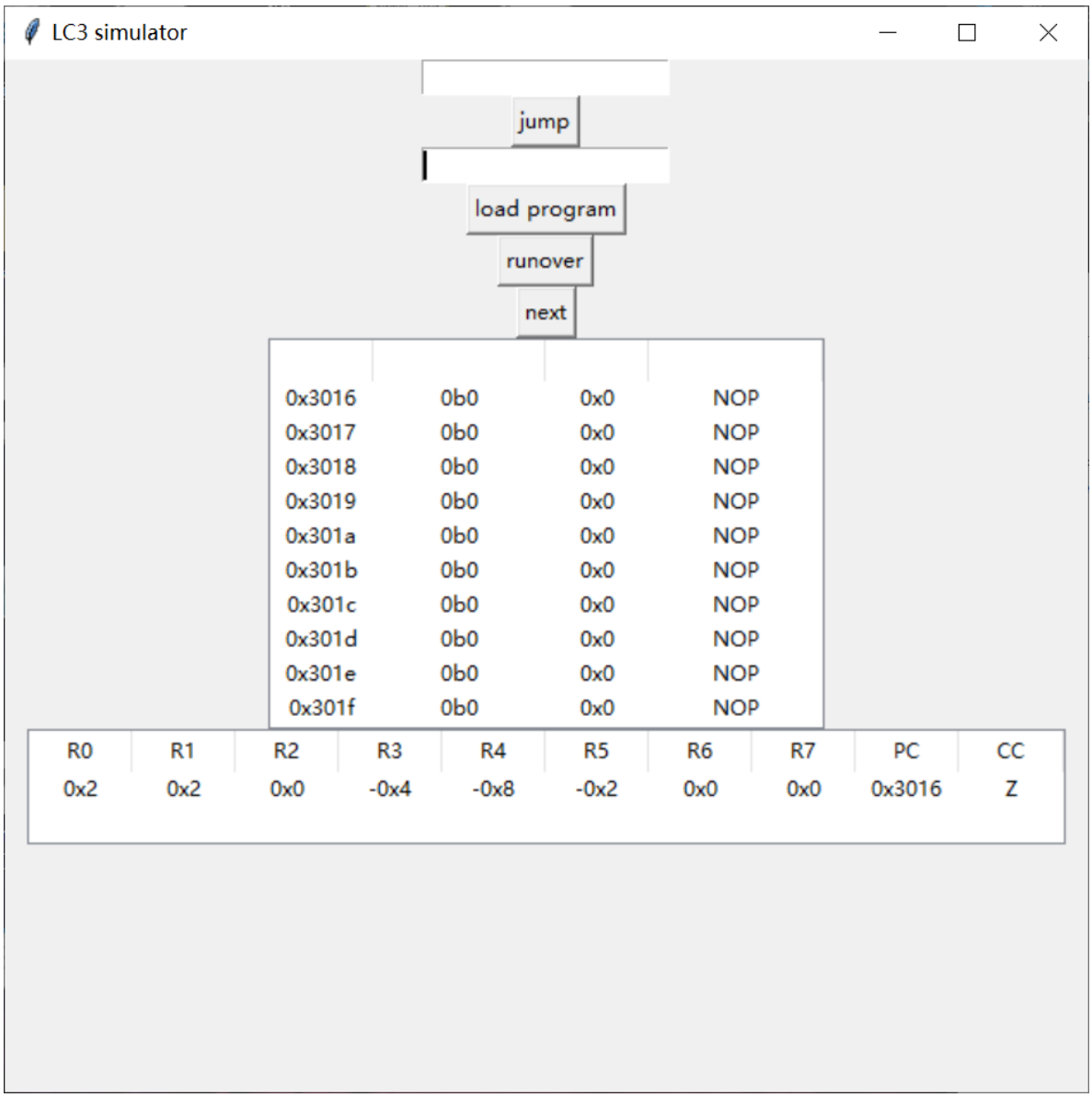
```
    global treeshow
    # Double-Button-1
    if(len(treeshow.selection())>1):
        print("select more than 1 row")
        return
    else:
        Show(event)

treeshow.bind('<Double-3>',SetShow)
```

展示截图




LC3 simulator
jump
R0 6
load program
runover
next
0x3000 0b10010100001 0x943f NOT R2, R0
0x3001 0b10100101000 0x14a1 ADD R2, R2, #1
0x3002 0b10100100000 0x1481 ADD R2, R2, R1
0x3003 0b1000001100 0x20c BRP 0x3010
0x3004 0b10000010000 0x410 BRZ 0x3015
0x3005 0b10010110011 0x967f NOT R3, R1
0x3006 0b10110111000 0x16e1 ADD R3, R3, #1
0x3007 0b11000111000 0x18e0 ADD R4, R3, #0
0x3008 0b11001000001 0x1904 ADD R4, R4, R4
0x3009 0b11010000001 0x1a04 ADD R5, R0, R4
R0 R1 R2 R3 R4 R5 R6 R7 PC CC
0x6 0x0 0x0 0x0 0x0 0x0 0x0 0x0 0x3000


LC3 simulator
jump
R1 8
load program
runover
next
0x3000 0b10010100001 0x943f NOT R2, R0
0x3001 0b10100101000 0x14a1 ADD R2, R2, #1
0x3002 0b10100100000 0x1481 ADD R2, R2, R1
0x3003 0b1000001100 0x20c BRP 0x3010
0x3004 0b10000010000 0x410 BRZ 0x3015
0x3005 0b10010110011 0x967f NOT R3, R1
0x3006 0b10110111000 0x16e1 ADD R3, R3, #1
0x3007 0b11000111000 0x18e0 ADD R4, R3, #0
0x3008 0b11001000001 0x1904 ADD R4, R4, R4
0x3009 0b11010000001 0x1a04 ADD R5, R0, R4
R0 R1 R2 R3 R4 R5 R6 R7 PC CC
0x6 0x8 0x0 0x0 0x0 0x0 0x0 0x0 0x3000


LC3 simulator
jump
load program
runover
next
0x3016 0b0 0x0 NOP
0x3017 0b0 0x0 NOP
0x3018 0b0 0x0 NOP
0x3019 0b0 0x0 NOP
0x301a 0b0 0x0 NOP
0x301b 0b0 0x0 NOP
0x301c 0b0 0x0 NOP
0x301d 0b0 0x0 NOP
0x301e 0b0 0x0 NOP
0x301f 0b0 0x0 NOP
R0 R1 R2 R3 R4 R5 R6 R7 PC CC
0x2 0x2 0x0 -0x4 -0x8 -0x2 0x0 0x0 0x3016 Z