**深 圳 大 学 实 验 报 告**

|  |
| --- |
| **课程名称 计算机系统1**  **项目名称 LC-3仿真器安装和使用**  **学 院 计算机与软件学院**  **专 业 计算机科学与技术**  **指导教师 陈飞**  **报 告 人 刘俊楠 学号 2017303010**  **实验时间 2021/4/8**  **提交时间 2021/4/8** |

**教务处制**

# 一、实验目的与要求

（1）掌握处理器仿真工具LC-3软件的安装和使用方法。

（2）学会在LC-3仿真环境下编辑程序和转换成可执行目标程序的方法 。

（2）学会在LC-3仿真环境下运行和调试程序的方法 。

# 二、实验内容与方法

利用提供的安装软件包和软件使用说明文档，完成以下试验内容：

（1）安装LC-3仿真器

（2）利用LC3EDIT输入机器代码程序（0/1模式）并创建可执行目标程序。

（3）利用LC3EDIT输入机器代码程序（hex模式）并创建可执行目标程序。

（4）利用LC3EDIT输入汇编代码程序并创建可执行目标程序。

（5）利用仿真器运用对应目标程序。

（6）学习和掌握断点，单步执行等调试方法和手段。

# 三、实验步骤与过程

（依照实验内容，逐条撰写实验过程与实验所得结果：包括程序总体设计，核心数据结构及算法流程，调试过程。请附上核心代码，及注意格式排版的美观。实验提交时，以上为评分依据，请不删除本行）

实验环境：

硬件：Pc电脑、windows10系统

软件：LC3Edit、LC3Simulate

# 1、EX3.ASM:其本质是将内存中加载十个数相加。

.ORIG x3000 AND R1,R1,x0 ;清 R1, 用来存放运行总和 (R1=R1 AND 0)

AND R4,R4,x0 ;清 R4, 用来作为一个计数器 (R4=R4 AND 0)

ADD R4,R4,xA ; 用#10 装载 R4, 添加的次数 (R4=R4+10)

LEA R2,x0FC ; 装载数据的起始地址 (R2=x0FC地址指向的值)

LOOP LDR R3,R2,x0 ; 装载下一个数被相加 (R3=R2+offset(000000))

ADD R2,R2,x1 ; 递增指针 (R2=R2+1)

ADD R1,R1,R3 ; 添加下一个数到运行总和 (R1=R1+R3)

ADD R4,R4,x-1 ; 递减计数器 (R4=R4-1)

BRP LOOP ; 如果计数器不为零，继续做 (if R4>0 PC指向LOOP)

HALT ；停止 (exit)

.END

1-1输入以上代码进入LC3Edit后，点击保存为EX3.asm（图1-1），然后点击Translate->Assemble 生成可执行文件EX3.obj。

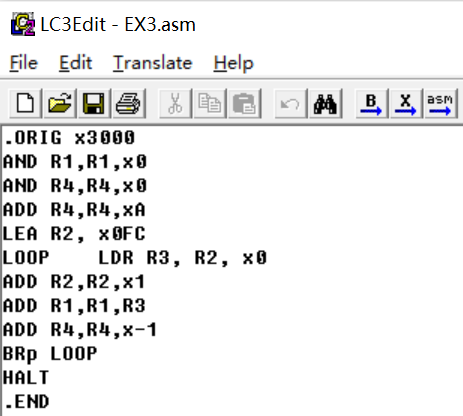


图1-1

1-2 打开LC3Simulate，运行EX3.obj（图1-2），可看到寄存器区（左上）、机器状态区（右上）、以及内存区（下）。发现内存区有十个内存有指令值，说明代码中第一行(.ORIG x3000)没有写入内存中，仅仅做了标记内存初始值。最后一行（.END）也仅仅是做结束标记。其余在内存中均有体现。R1-R7寄存器都为0值

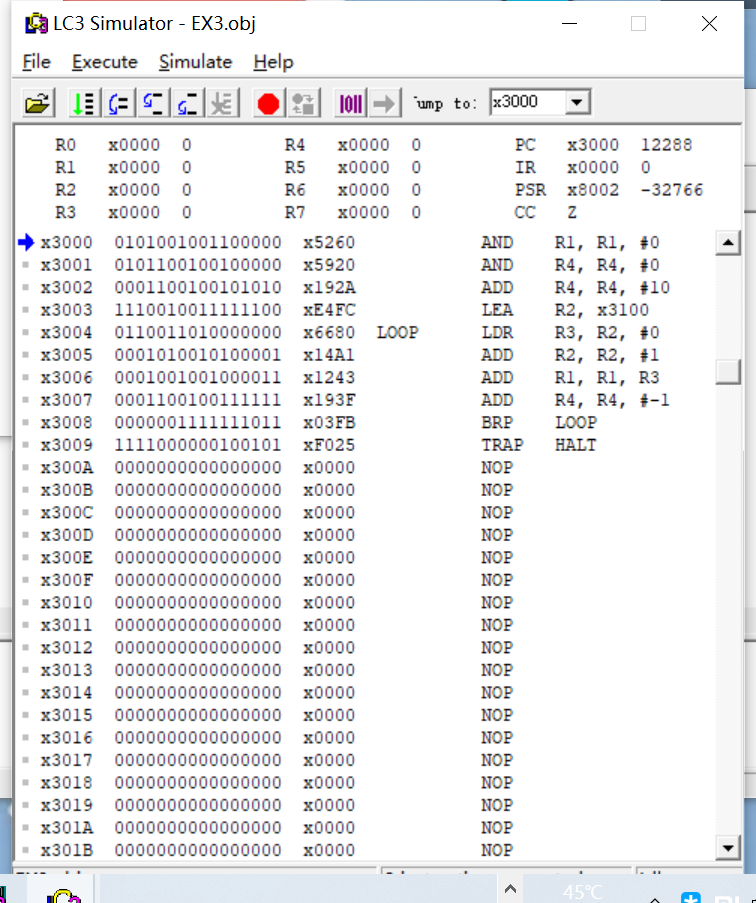


图1-2

1-3 加载数据（十个数字）到内存，返回到 LC3Edit.程序，输入下面 16 进制代码:

3100 ;数据在内存地址 X3100 处开始

3107 ;从此处开始，添加的 10 个数字 （X3101）

2819 (X3102)

0110 (X3103)

0310 …

0110

1110

11B1

0019

0007

0004

1-4保存为data.hex后，生成可执行文件data.obj，并在打开EX3.obj的LC3Simulate中导入该文件，可看到x3100-x3109均被赋予了值（图1-4）：

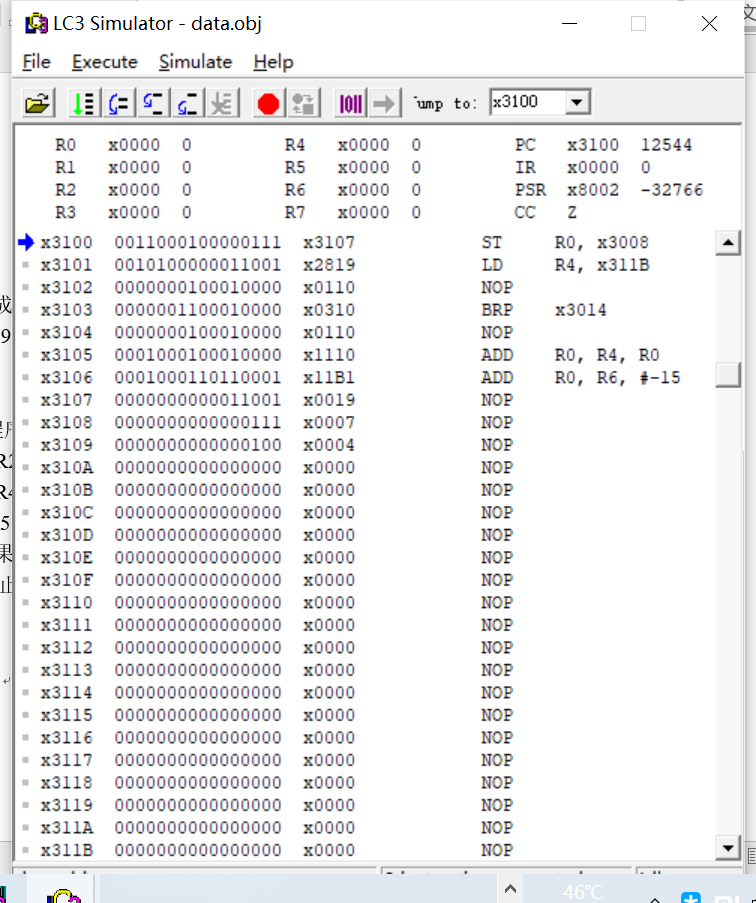


图1-4

1-5 jump至x3000处，并且PC指向x3000，在x3009 处TRAP的位置设置断点，单步执行程序。由图1-5-1可知，PC指向下一执行指令内存，也就是x3001，IR指向运行完毕指令的值，也就是x3000的值x5260。在执行到3009时，得到目标值PC=x3009 R1=x8135（图1-5-2），与书上值相差x10，计算机计算得知我为正确答案。书上写错了。

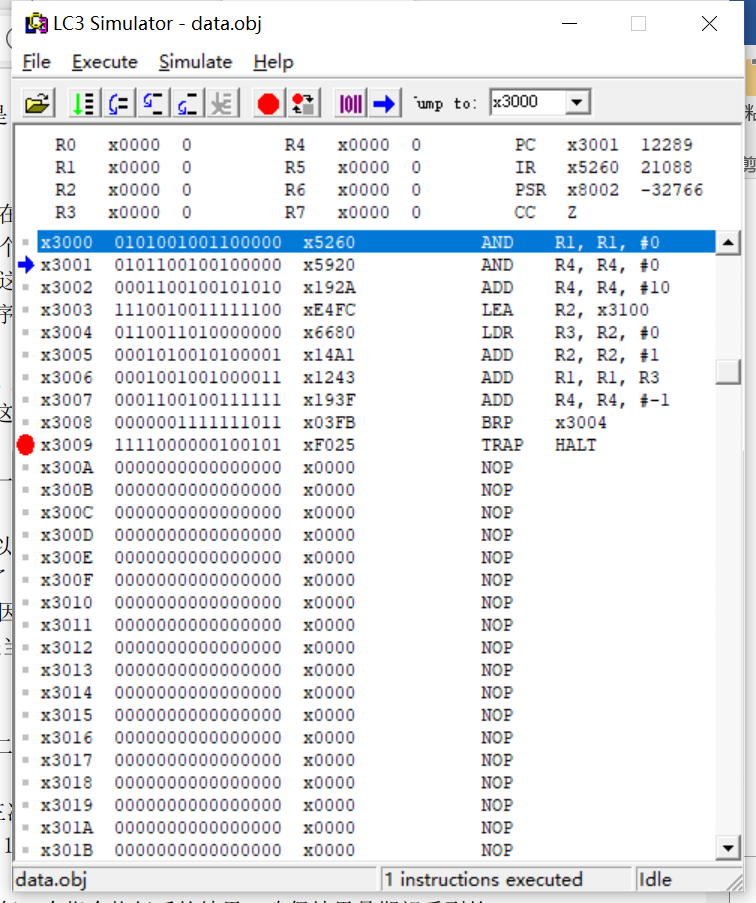


图1-5-1

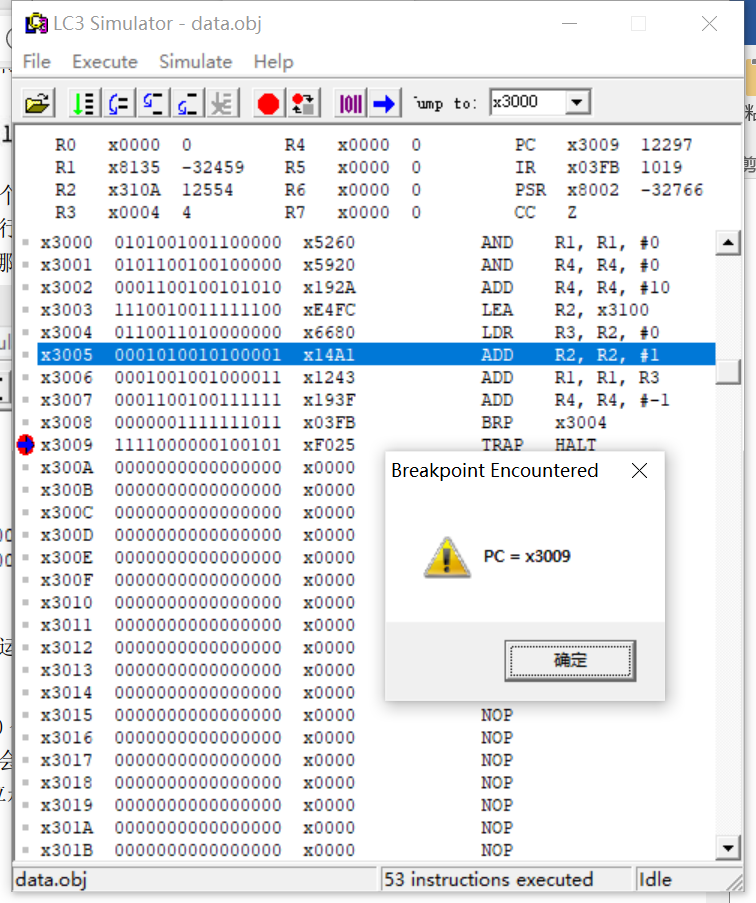


图1-5-2



图1-5-3

# 2、Example1:本例实现通过加法实现乘法，在寄存器中实现为R2=R4\*R5。

0011 0010 0000 0000 ;程序起始地址: x3200

0101 010 010 1 00000 ; R2 复位 (R2=R2 AND 0)

0001 010 010 0 00 100 ;R4中值与R2相加 结果放置与R2中 (R2=R2+R4)

0001 101 101 1 11111 ;R5中值减去1 (R5=R5-1)

0000 011 111111101 ;如果结果>=0 转移至x3201 (if R5>0 PC指向x3201)

1111 0000 00100101 ;停止 (exit)

2-1 保存为Example1.bin 并且生成Example1.obj可执行文件后，在模拟器中运行，并在x3204(TRAP处)设置断点，双击R4与R5的值，分别设置为x5与x3并运行程序（图2-1-1）。此时看到目标值R2为20，与3\*5=15不相符，于是开始调试。

在x3203处设置断点后，点击运行程序，第一次R2返回5，正确，第二次返回10，正确，第三次返回15，此时机器状态栏中的CC应该为P（图2-1-2），此时却为Z,说明还要循环一次，故发现问题。

经过查阅代码发现，循环结束时条件语句的条件多考虑了一次 应该调整为>0而不是>=0。调整后进入模拟器运行，结果正确，调试成功。（图2-1-3）

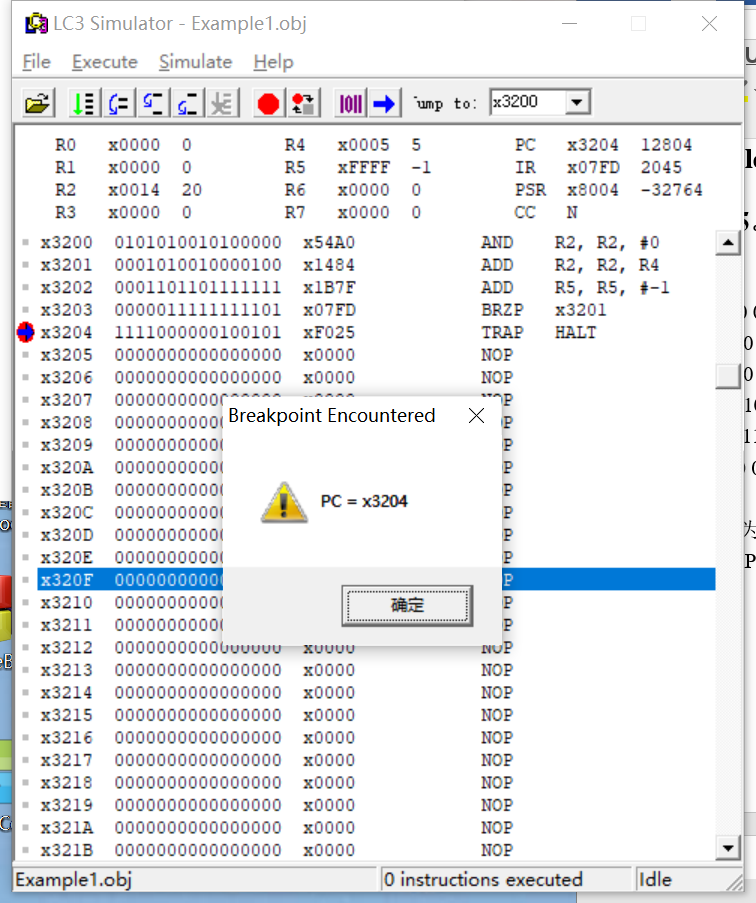


图2-1-1

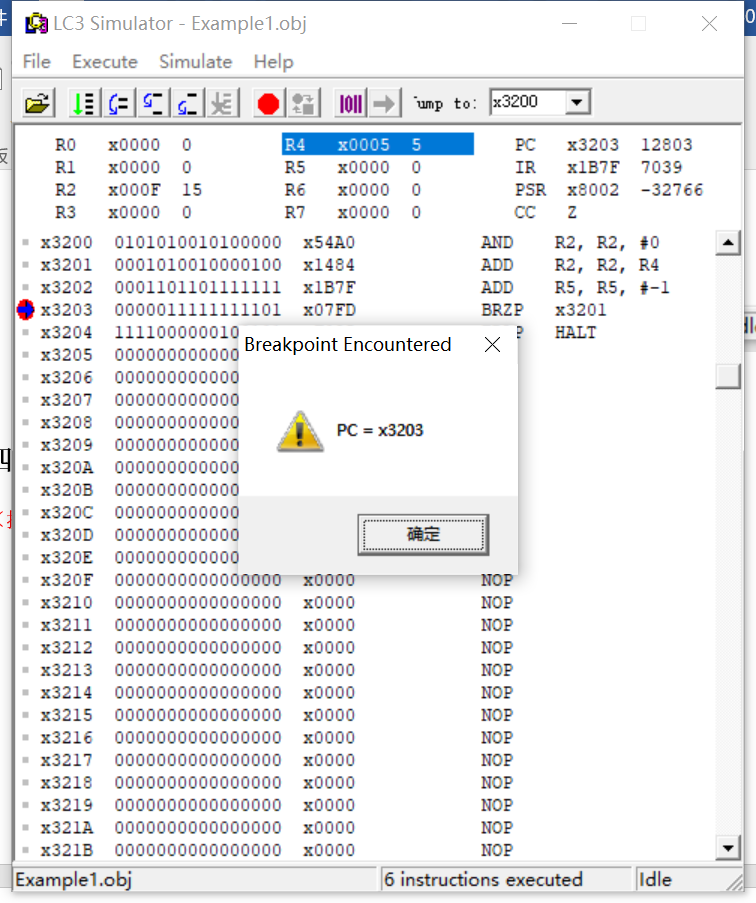


图2-1-2

# 3、Example3:求两个输入的和。

.ORIG x3000 (设置起始地址x3000)

TRAP x23 ;the trap instruction which is also known as "IN" (输入值赋值给R0)

ADD R1,R0,x0 ;move the first integer to register 1 (R1=R0+0)

TRAP x23 ;another "IN" (输入值赋值给 R0)

ADD R2，R0，R1 ;两个整数相加 (R2=R1+R0)

LEA R0，MESG ;载入字符串的地址 (R0=MESG地址指向的值)

TRAP x22 ;输出字符串 (printf(MESG指向的字符串))

ADD R0，R2，x0 ;sum 保存到 R0 中，并准备输出 (R0=R2+0)

TRAP x21 ;显示结果 (printf(R0))

HALT MESG .STRINGZ “The sum of those two numbers is” (MESG指向字符串)

.END (exit)

3-1 将上述代码保存为Example2.asm,并生成可执行文件Example2.obj,在模拟器中运行。但在编译时出现问题，其指出不能用中文字符，故修改代码（图3-1-1）。修改其中的“，”“。”之后，编译成功，生成目标文件Example2.obj（图3-1-2）。在模拟器运行，输入4与4后，输出为h，意识到代码错误（图3-1-3）。查Ascall表知，4在表中为x34，h在表中为x68，所以明白是进行了ascall码的加减，而不是进行数字加减，应该先减x30，相加之后在加上x30，故改正成如下代码（3-1-4）。改正后在模拟器运行，结果正确，调试成功。

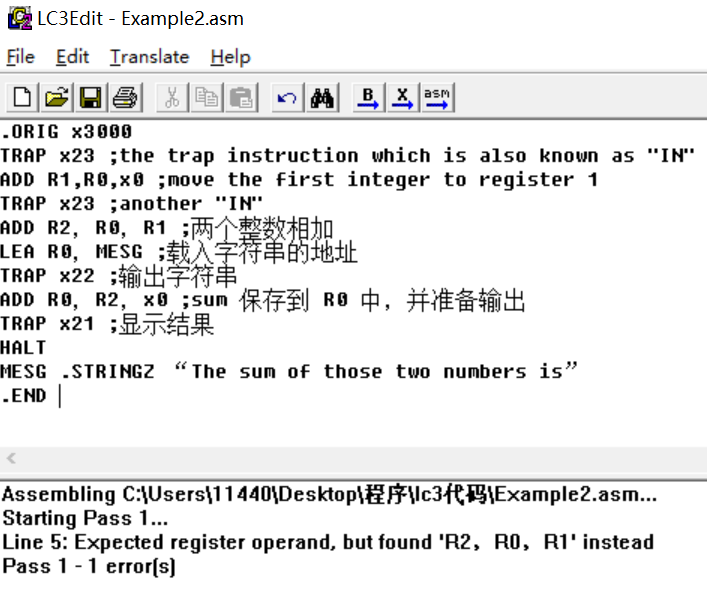


图3-1-1

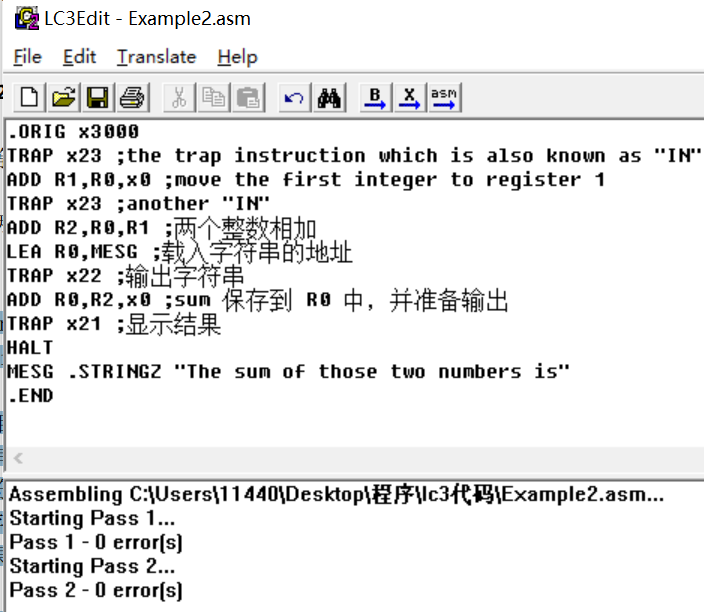


图3-1-2

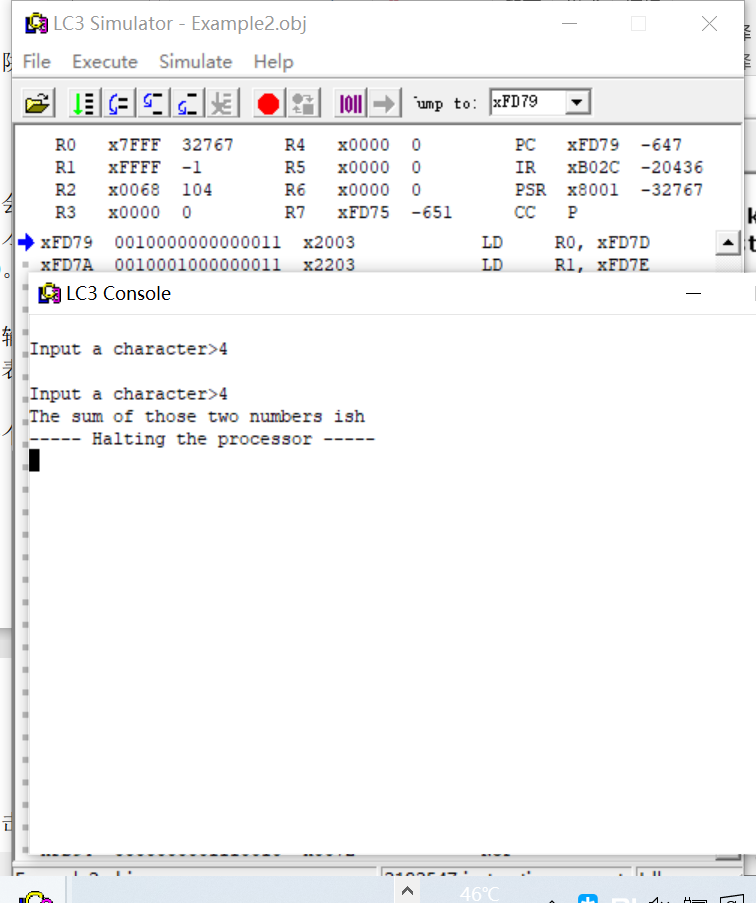


图3-1-3

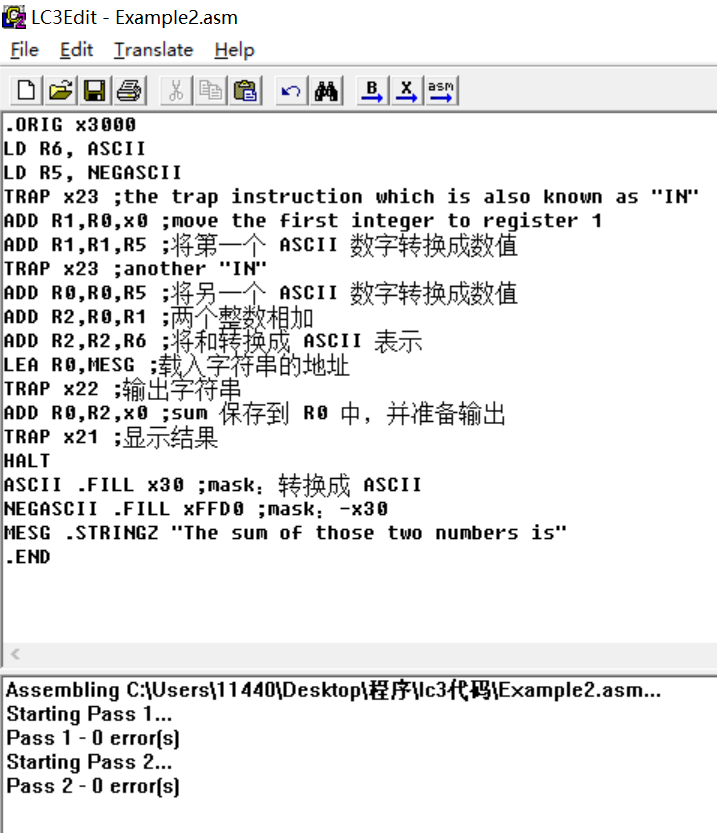


图3-1-4

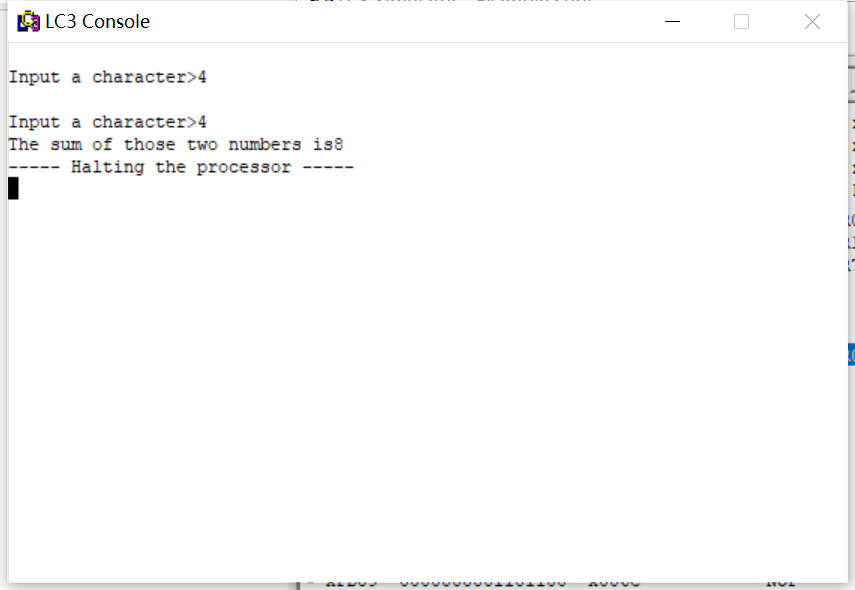


图3-1-5

# 四、实验结论或体会

（撰写实验收获及思考）

结论：

通过本次实验，我学会了如何安装LC-3仿真器，并且利用LC3EDIT软件输入机器代码程序（0/1模式）保存为二进制文件并创建可执行obj目标程序。利用LC3EDIT软件输入机器代码程序（hex模式）保存为十六进制文件并创建可执行obj目标程序。利用LC3EDIT输入汇编代码asm程序并创建可执行sym目标程序。利用仿真器运用对应目标程序。学习和掌握了断点，单步执行等调试方法和手段。

思考：

为什么data.hex中的十个数据在模拟器中会有各种运算符参与？

答：在冯诺依曼计算机下，不区分数据、指令，事实上，在 LC-3simulator 眼中所有内存位置的内容都是一样的。除非被告知 作为指令运行，或者作为数据加载，所以，这些运算符并没有什么意义。

|  |
| --- |
| 指导教师批阅意见：  成绩评定：  指导教师签字：  年 月 日 |
| 备注： |

注：1、报告内的项目或内容设置，可根据实际情况加以调整和补充。

2、教师批改学生实验报告时间应在学生提交实验报告时间后10日内。