

# lc3 tools使用指南

---

## 一、下载安装

课程主页上为较老版本的软件，更推荐下载最新版2.0.2版本的软件。

建议通过以下两种方式进行下载：

- github下载
- 蓝奏云下载

若网络条件允许，更推荐github下载。

观前提醒：

请保证软件的路径全英文，不允许有中文！

请保证软件的路径全英文，不允许有中文！














请保证软件的路径全英文，不允许有中文！

### 1.github下载（推荐）

如果网络条件允许的话，更推荐这种方式。如果网络条件不允许，可以选择第二种方式。

链接为：<https://github.com/chiragsakhuja/lc3tools/releases/tag/v2.0.2>

根据自己的操作系统版本点击下载即可。（例如：windows选择LC3Tools-2.0.2.exe，macOS选择LC3Tools-2.0.2.dmg）

▼ Assets 11		
 latest-linux.yml	366 Bytes	15 Nov 2021
 latest-mac.yml	332 Bytes	15 Nov 2021
 latest.yml	344 Bytes	15 Nov 2021
 LC3Tools-2.0.2.ApplImage	64.8 MB	15 Nov 2021
 LC3Tools-2.0.2.dmg	62.1 MB	15 Nov 2021
 LC3Tools-2.0.2.dmg.blockmap	67.9 KB	15 Nov 2021
 LC3Tools-2.0.2.exe	54.5 MB	15 Nov 2021
 LC3Tools-Setup-2.0.2.exe	54.6 MB	15 Nov 2021
 LC3Tools-Setup-2.0.2.exe.blockmap	58.9 KB	15 Nov 2021
 <a href="#">Source code</a> (zip)		15 Nov 2021
 <a href="#">Source code</a> (tar.gz)		15 Nov 2021
  2 2 people reacted		

## 2. 蓝奏云下载

考虑到部分同学的网络环境，不一定能上github。故提供网盘下载。（无需登录，下载满速）

链接: <https://wwm.lanzouy.com/b03p95ytg>

密码:5iya

根据自己的操作系统版本点击下载即可。（例如: windows选择LC3Tools-2.0.2.exe）

## 二、打开使用

由于助教是windows系统，所以下面以windows版使用为例。

再次提醒:

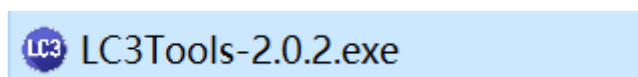
请保证软件的路径全英文，不允许有中文！

请保证软件的路径全英文，不允许有中文！

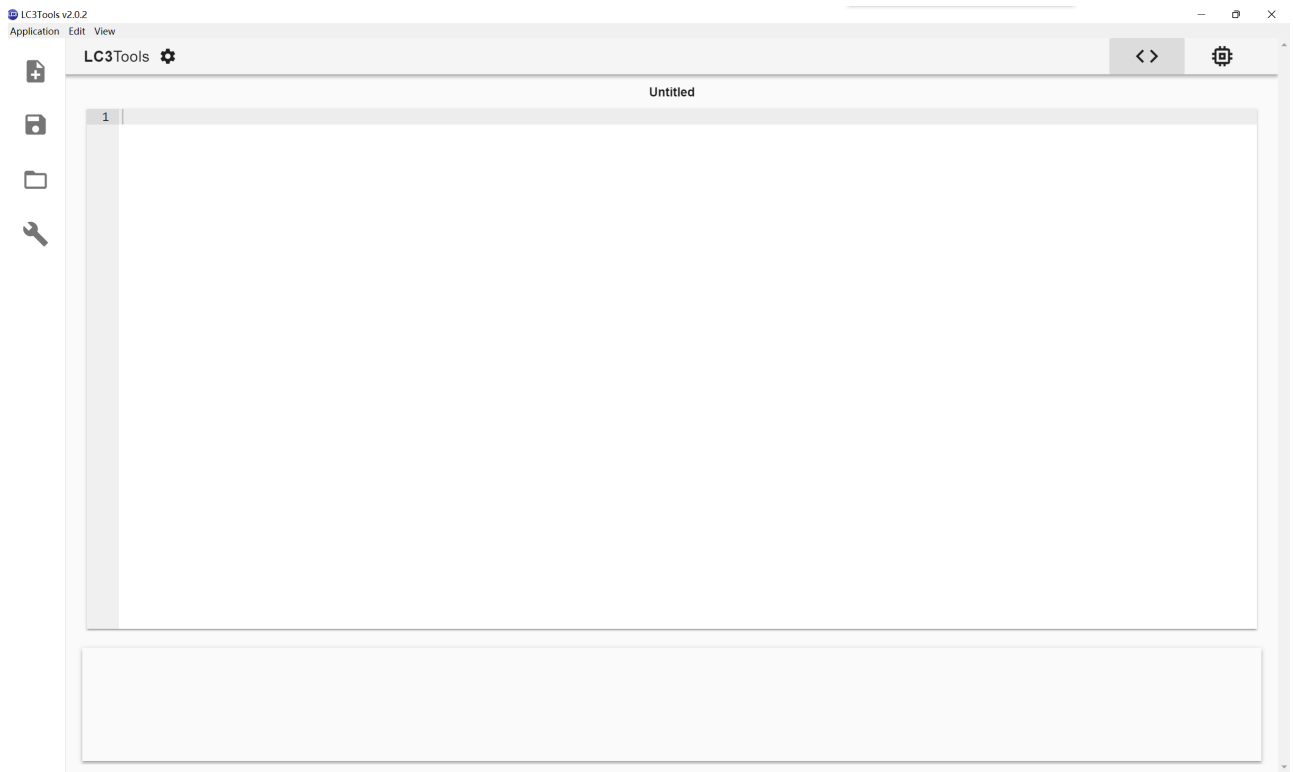
请保证软件的路径全英文，不允许有中文！

### 1.打开软件

文件夹下视图如下:



直接双击打开即可。打开后便能看到如下界面:

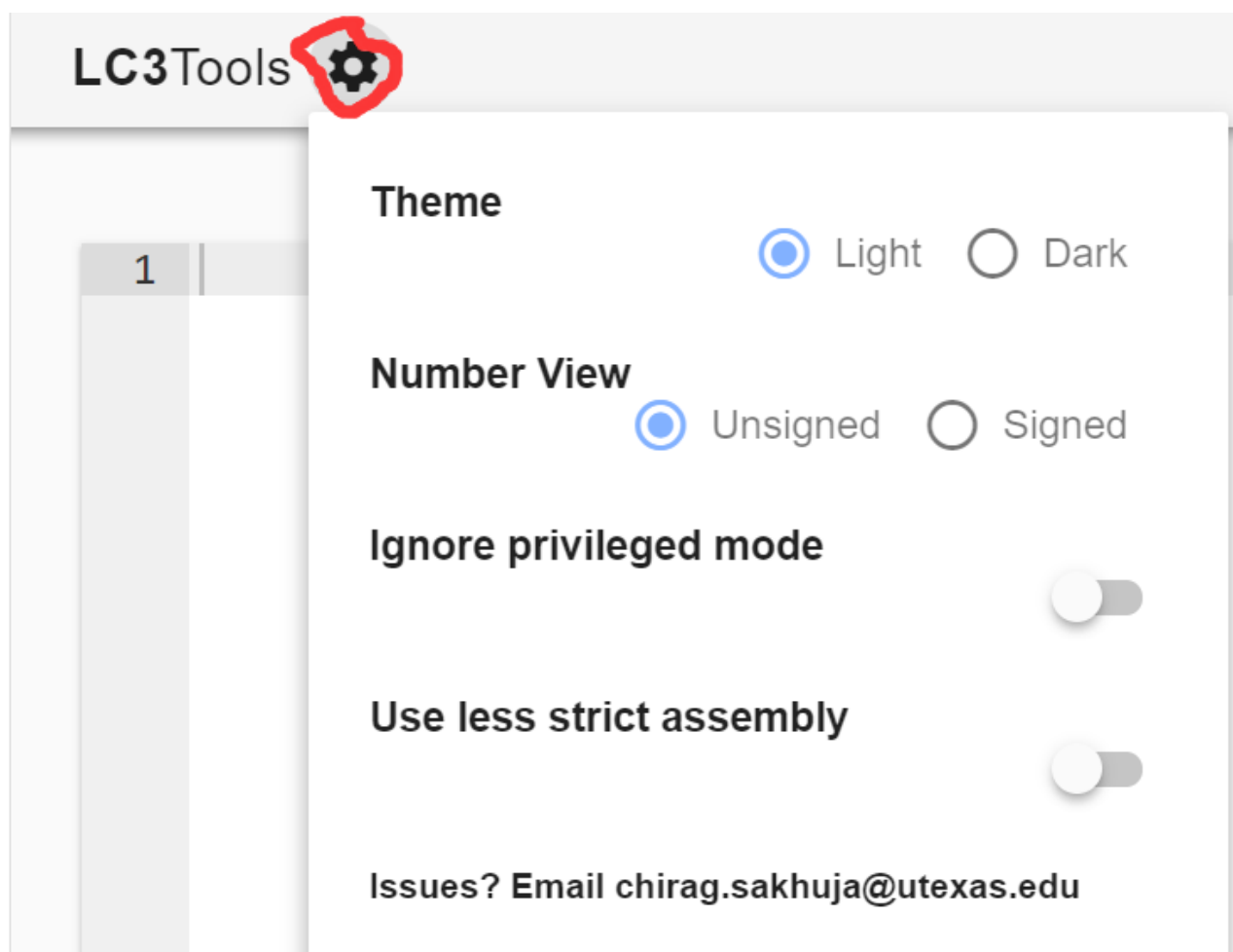


## 2.简单设置

### I.主题

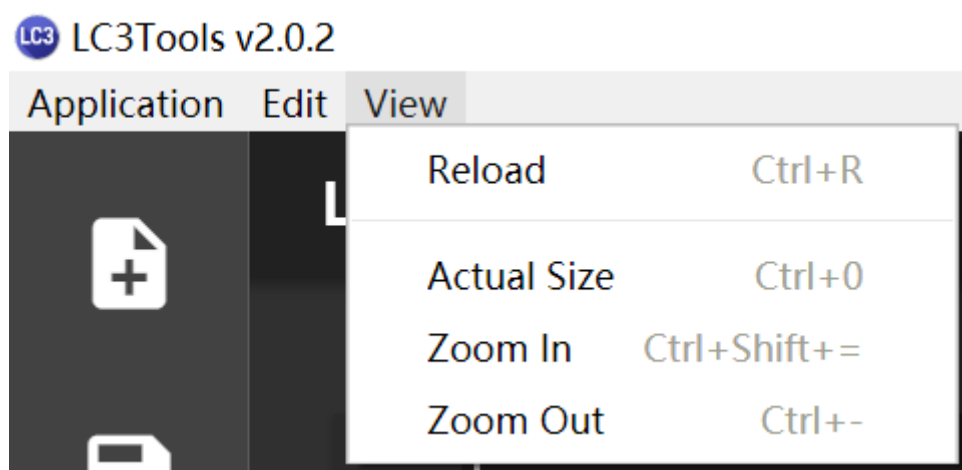
你可能会觉得白色的界面太丑，想切换成黑色？

点击左上角的齿轮，切换为黑色主题！



## II. 界面大小

你可能觉得字体太小或太大，可以点击左上角的View菜单，通过Zoom In和Zoom Out来调整大小。



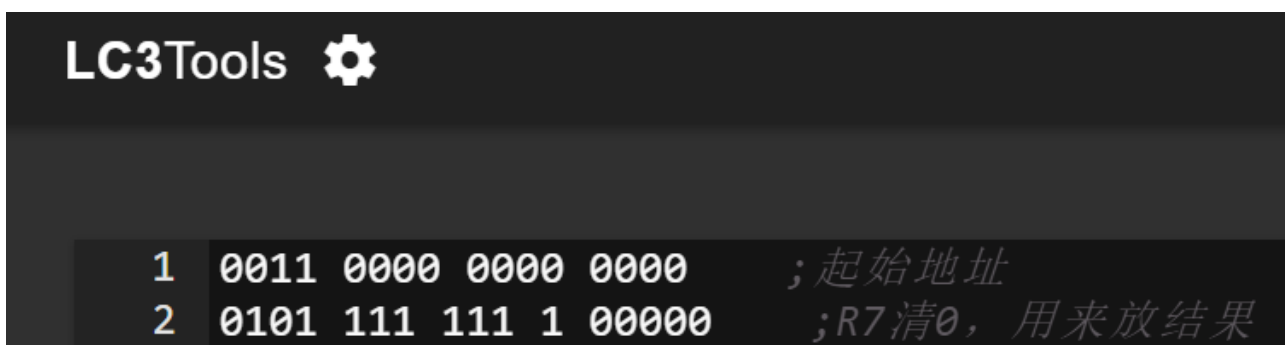
## 3. 认识编辑界面

首先可以看下图，主要分为三个区域，下面将分别讲解。



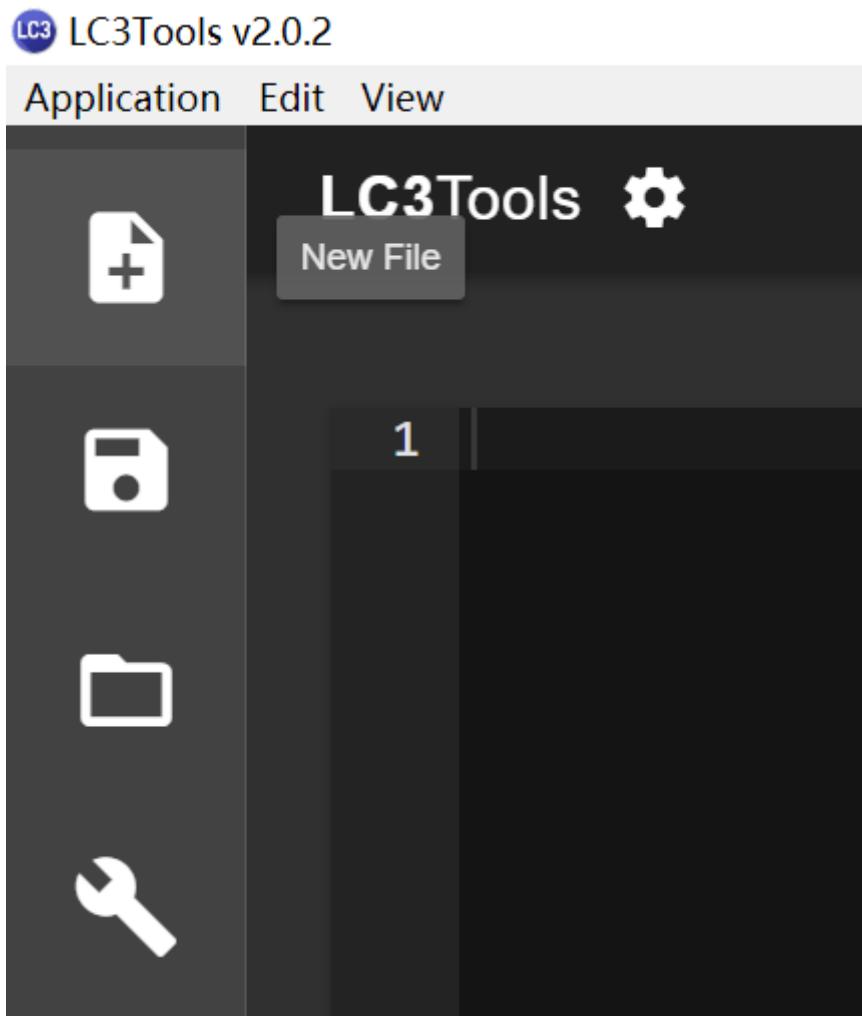
## I. 编辑区

最中间的就是代码编辑区，你可以在里面输入机器码或者汇编码，如下所示：（你可以暂时不管下图中机器码的意思）



## II. 功能区

然后左边的四个按钮，我们称其为“功能区”。将鼠标移到图标上，会看到说明：

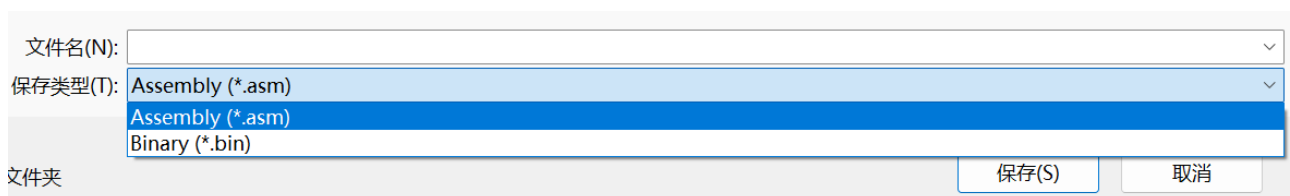


四个按键功能依次为：创建新文件，保存文件，打开文件，生成.obj文件。

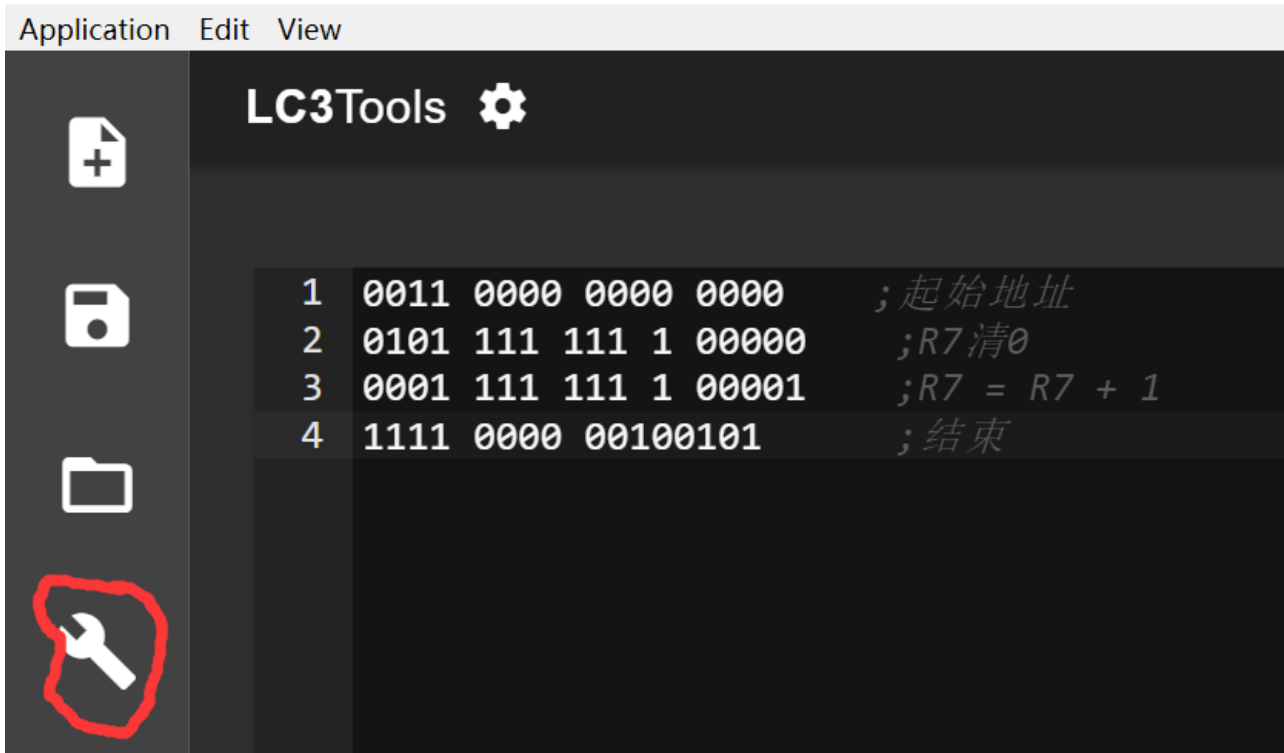
值得一提的是第一个和第四个。

在创建新文件过程中，如果想写机器码文件（01码），请选择xxx.bin。如果想写汇编语言文件，请选择xxx.asm。

第一个实验是写机器码，后几个实验是写汇编。



第四个按键，用于将机器码或汇编码转成.obj文件。例如下面一个机器代码：

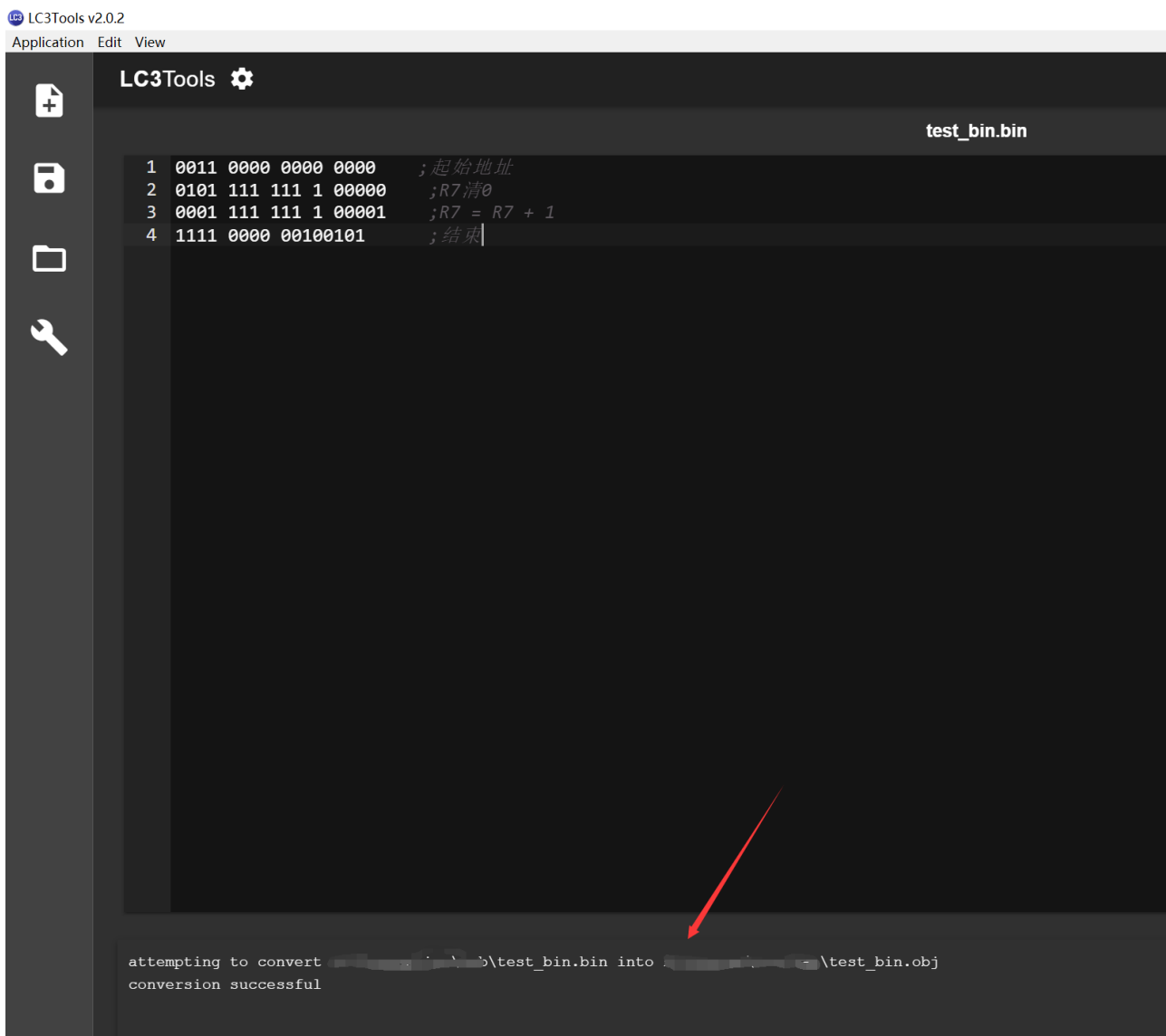


你可以直接复制以下代码，到你的软件中进行尝试：

```
0011 0000 0000 0000 ;起始地址
0101 111 111 1 00000 ;R7清0
0001 111 111 1 00001 ;R7 = R7 + 1
1111 0000 00100101 ;结束
```

点击“扳手”后，会看到下面提示转化为.obj文件，且转化成功：

当然，如果转化出错，最下方也会报错。

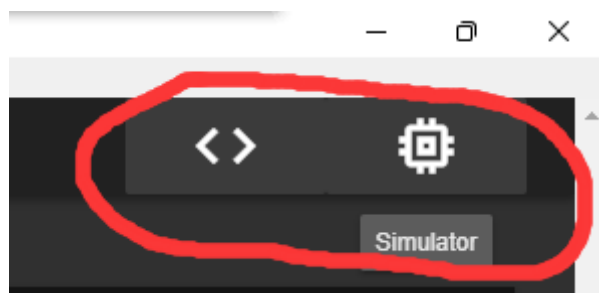


### III. 界面切换

最后来认识以下右上角的按钮。

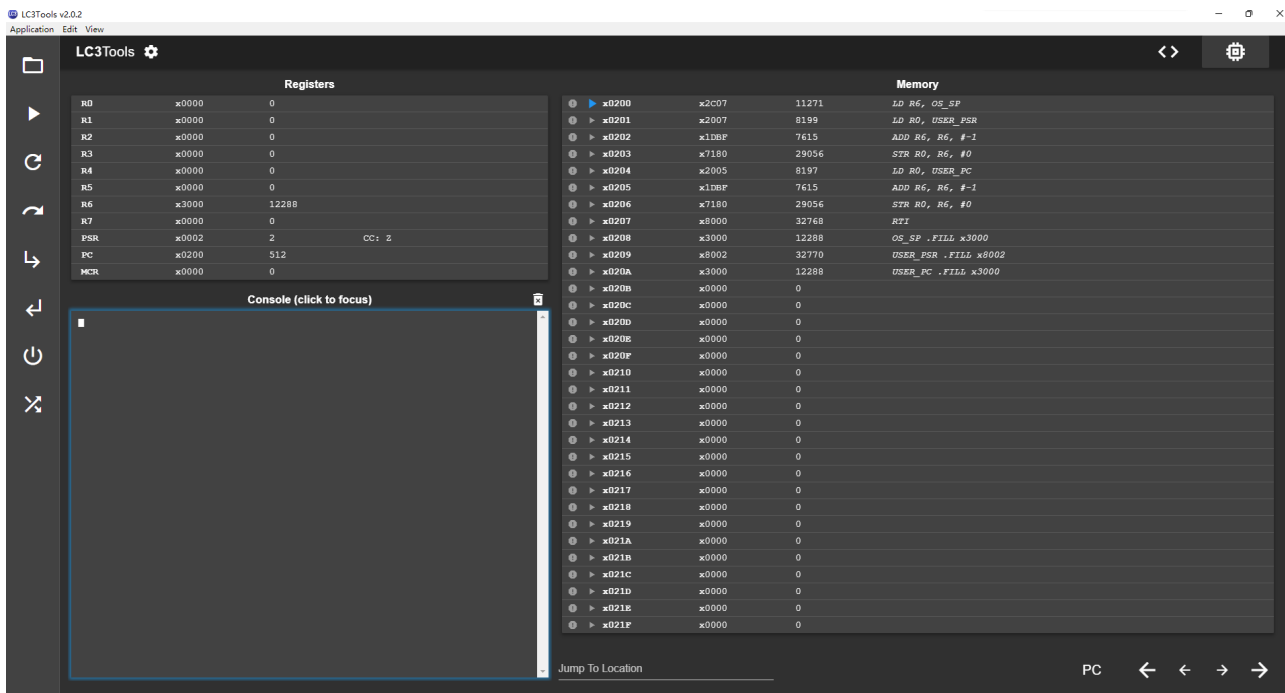
左边的即当前界面，点击右边，可以进入一个新的界面，我们称其为模拟区。

当在左边代码编辑区写完代码并生成obj之后，转到这个界面就能运行刚刚写的程序。



模拟区如下图所示，我们将在下一个环节进行介绍：





## 4.认识模拟界面

在我们写完代码后，当然是要跑出结果啦。

而模拟界面，则是用上一步得到的.obj文件，跑出代码运行结果。

模拟界面可以分为四个核心区域，即最左边的功能区、左上角的寄存器区、左下角的终端区、右边的内存区。

下面将逐个介绍：

### I.寄存器区

正如下图所示，寄存器区显示了每个寄存器的信息。有八个通用寄存器（R0-R7，你可以简单理解成变量）和PSR、PC、MCR。

第一列是寄存器名字，第三列是寄存器的值（十进制），第二列也是寄存器的值（十六进制）。

Registers			
R0	x0000	0	
R1	x0000	0	
R2	x0000	0	
R3	x0000	0	
R4	x0000	0	
R5	x0000	0	
R6	x3000	12288	
R7	x0000	0	
PSR	x0002	2	CC: Z
PC	x0200	512	
MCR	x0000	0	

## II.终端区

会用于一些信息的提示，如程序运行结束的提示。

以后有可能会用于程序的键盘输入输出，现阶段不再赘述。



### III.内存区

当程序在运行的时候，指令和部分数据会加载到内存中。而内存区，顾名思义，就是展示内存的信息。

Memory				
❗	▶ x0200	x2C07	11271	LD R6, OS_SP
❗	▶ x0201	x2007	8199	LD R0, USER_PSR
❗	▶ x0202	x1DBF	7615	ADD R6, R6, #-1
❗	▶ x0203	x7180	29056	STR R0, R6, #0
❗	▶ x0204	x2005	8197	LD R0, USER_PC
❗	▶ x0205	x1DBF	7615	ADD R6, R6, #-1
❗	▶ x0206	x7180	29056	STR R0, R6, #0
❗	▶ x0207	x8000	32768	RTI
❗	▶ x0208	x3000	12288	OS_SP .FILL x3000
❗	▶ x0209	x8002	32770	USER_PSR .FILL x8002
❗	▶ x020A	x3000	12288	USER_PC .FILL x3000
❗	▶ x020B	x0000	0	
❗	▶ x020C	x0000	0	
❗	▶ x020D	x0000	0	
❗	▶ x020E	x0000	0	
❗	▶ x020F	x0000	0	
❗	▶ x0210	x0000	0	
❗	▶ x0211	x0000	0	
❗	▶ x0212	x0000	0	
❗	▶ x0213	x0000	0	
❗	▶ x0214	x0000	0	
❗	▶ x0215	x0000	0	
❗	▶ x0216	x0000	0	
❗	▶ x0217	x0000	0	
❗	▶ x0218	x0000	0	
❗	▶ x0219	x0000	0	
❗	▶ x021A	x0000	0	
❗	▶ x021B	x0000	0	
❗	▶ x021C	x0000	0	
❗	▶ x021D	x0000	0	
❗	▶ x021E	x0000	0	
❗	▶ x021F	x0000	0	

最左边这一列感叹号，点一下会变红❗

相当于其他IDE中“打断点”的功能。程序运行到这一行，会暂停。这功能有助于debug。

而第二列三角形符号，当程序即将要运行到某一行时，这一行的箭头会变色。

第三列，则是内存地址的十六进制表示。

第四列，则是内存里存的值的十六进制表示。

第五列，则是内存里存的值的十进制表示。

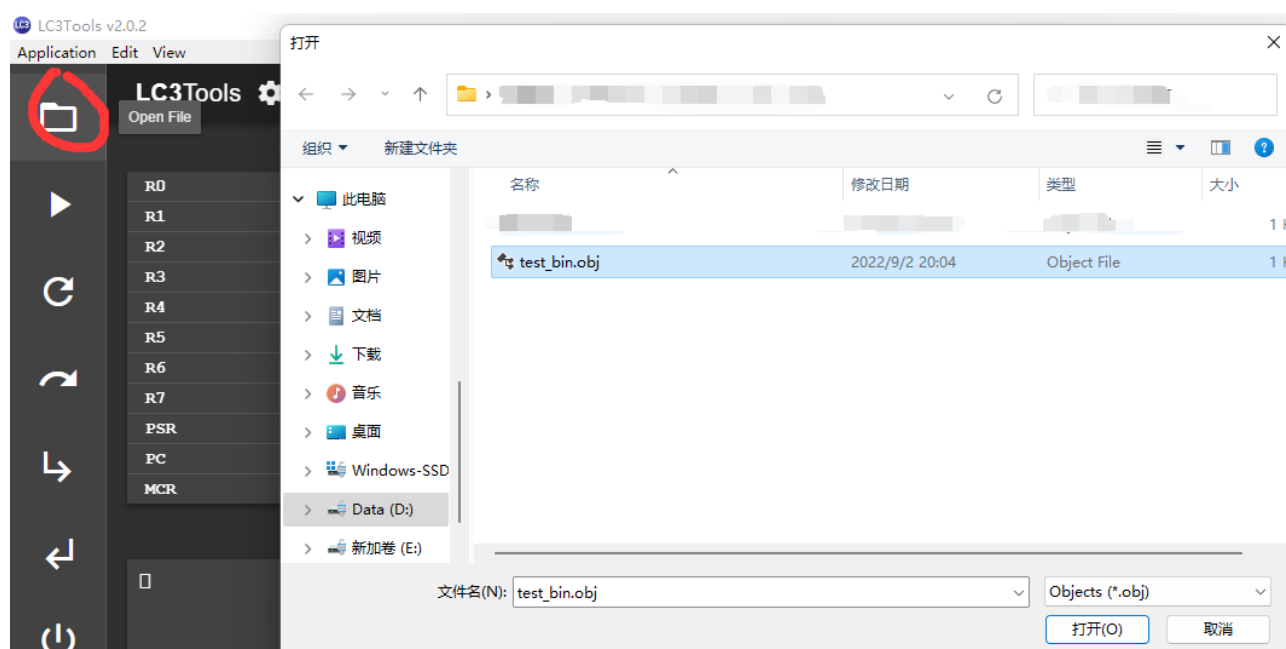
第六列，也是最后一列，则是指令的汇编或机器码。

## IV.功能区

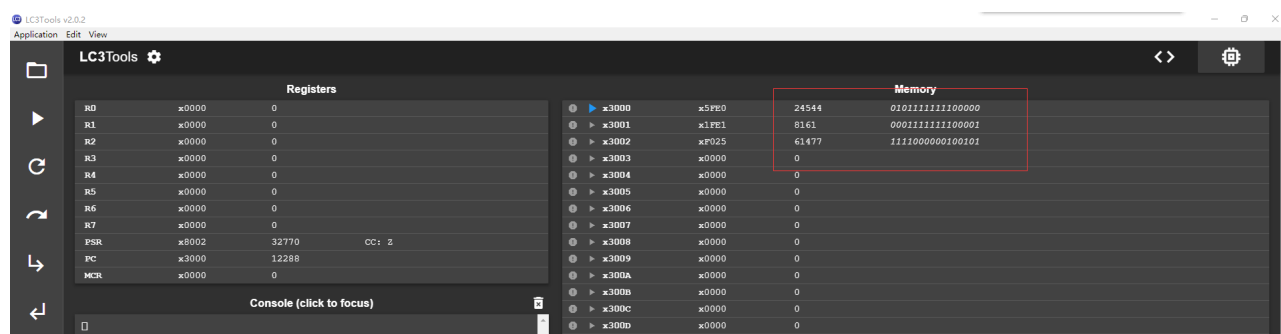
与编辑区类似的，功能区有多个按键，从上到下依次是：

- 打开文件
- 运行代码
- 重新载入代码
- 单步执行
- 步进
- 步出
- 重新初始化机器状态
- 随机生成机器状态

使用模拟界面前，必须先打开文件，选择刚刚生成的.obj文件：

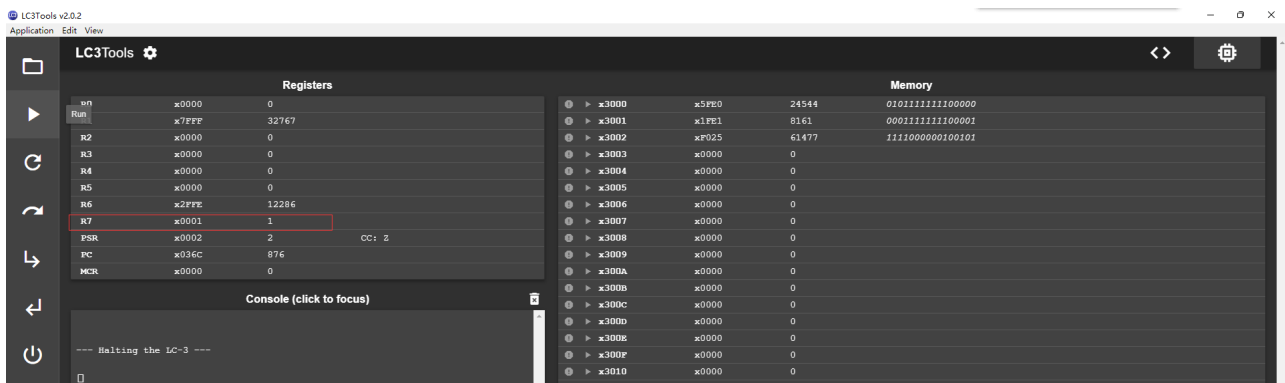


打开后会发现内存中已经载入刚才写的代码：



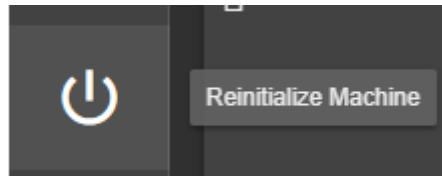
然后我们可以点击“Run”来运行一下我们的代码。（我们的代码功能是将R7清零后加一，因此正确结果应该是R7值为1）

刷了一下，很快嘛，我们的代码就跑完了，可以看到我们的R7值的确为1。

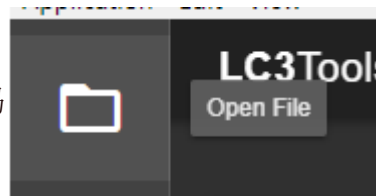


当然，这太快了，我们都没看清楚，还是一起来看看单步执行的效果吧。

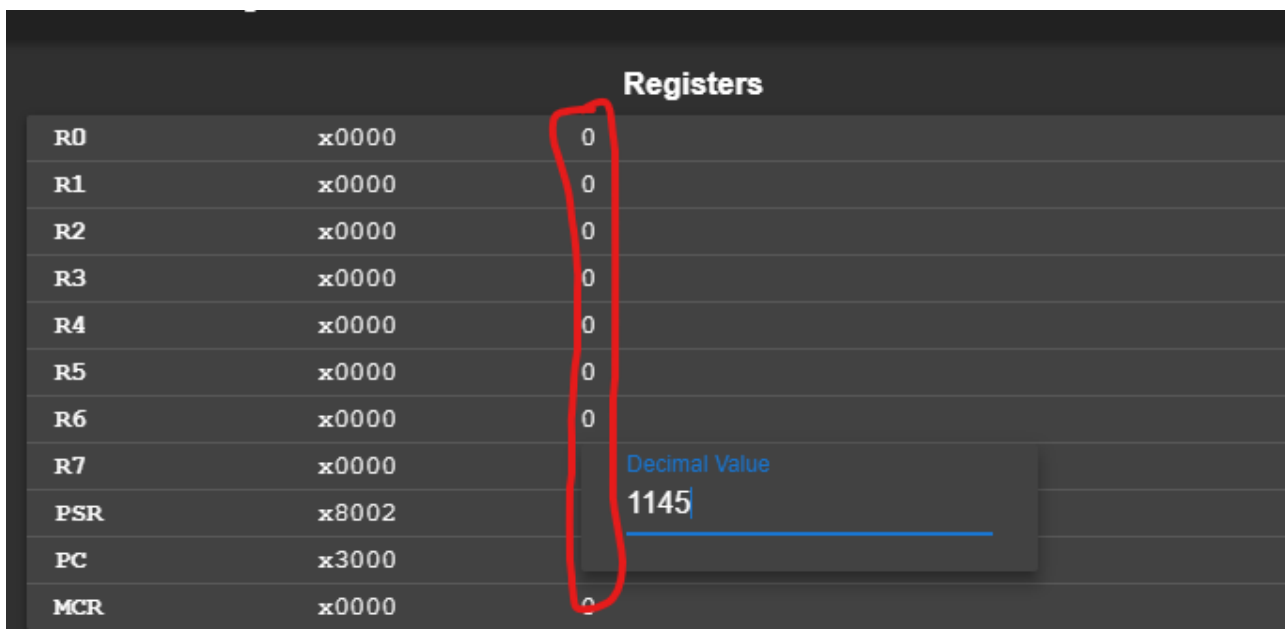
在此之前，先点击重新初始化按钮



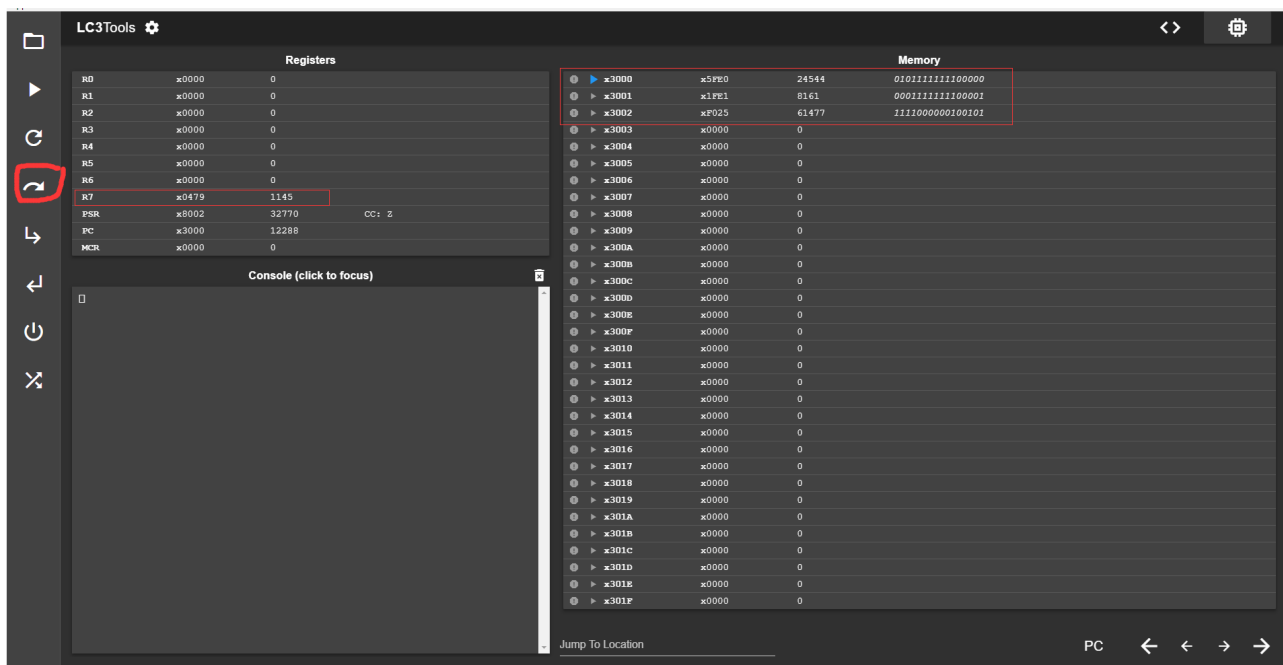
然后再重新载入代码



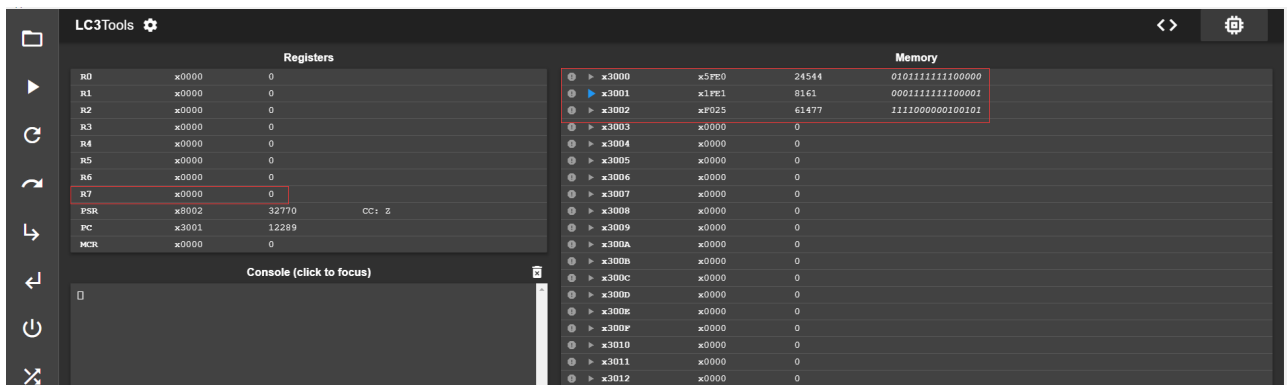
为了检验“清零R7”这一行代码，我们可以手动输入一个非零的R7值。输入也很简单，直接点击寄存器区的数值就好了。



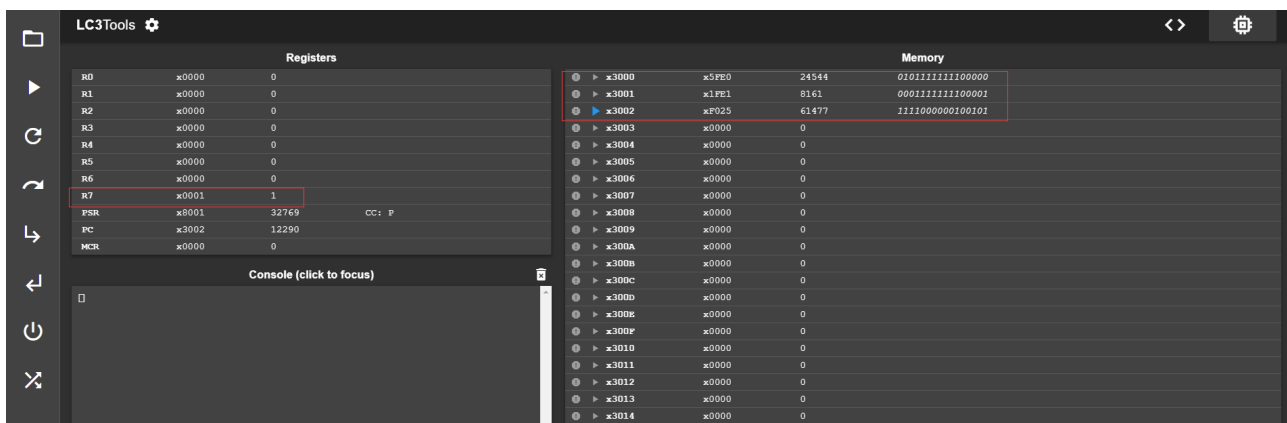
然后我们的初始状态就长这样：



下面我们先点击一次单步执行，可以看到R7的确被清零了：



让我们再点一次单步执行，可以看到R7的值的的确变成1了：



最后再点一次单步执行，结束程序。

### 三、简单总结

最后，来总结一下怎么使用lc3 tools吧。

再次提醒：

请保证软件的路径全英文，不允许有中文！

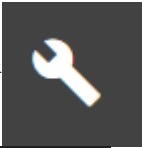
请保证软件的路径全英文，不允许有中文！

请保证软件的路径全英文，不允许有中文！

1. 在编辑区敲代码：

```
test_bin.bin
1 0011 0000 0000 0000 ;起始地址
2 0101 111 111 1 00000 ;R7清0
3 0001 111 111 1 00001 ;R7 = R7 + 1
4 1111 0000 00100101 ;结束
```

2. 点击扳手，生成.obj文件



3. 切换到模拟器界面



4. 在模拟器界面打开.obj文件



5. 在寄存器区自定义初始状态（可有可无，看你需求）

Registers			
R0	x0000	0	
R1	x7FFF	32767	
R2	x0000	0	
R3	x0000	0	
R4	x0000	0	
R5	x0000	0	
R6	x2FFE	12286	
R7	x0001	1	
PSR	x0002	2	CC: Z
PC	x036C	876	
MCR	x0000	0	

6. 运行代码



7. 如果要重复运行，可以选择



或



重新载入代码。