第一次实验

姓名: 傅申 学号: PB20000051

1 L 版本程序设计
1.1 理论
1.2 L 版本程序
1.2.1 最初版本
1.2.2 优化与最终版本
2 P 版本程序设计

1 L 版本程序设计

1.1 理论

记 a 和 b 的乘积为 $MUL(a,b)$, 则当 $a \geq 0$ 时, 有

$$MUL(a,b) = \begin{cases} 0 & a = 0 \\ b + MUL(a-1,b) & a \neq 0 \end{cases} \tag{1}$$

当 $a < 0$ 时, 考虑到 LC3 的寄存器是 16 位的, 所以 a 的补码为 $2^{16} + a$, 则当 a 减去了 $2^{16} + a$ 次 1 后, a 在寄存器中就变为了 `x0000`, 也就是 0. 按照上式计算, 有

$$MUL(a,b) \equiv ab \equiv 2^{16} + ab (\bmod 2^{16})$$

不考虑溢出 $(-2^{15} \leq ab \leq 2^{15} - 1)$, 有:

- $a \geq 0$ 时 $MUL(a,b) = ab$;
- $a < 0$ 时:
  - 若 $b \leq 0 \Rightarrow ab \geq 0$, 则 $MUL(a,b)$ 在寄存器中就是 ab;
  - 若 $b > 0 \Rightarrow ab < 0$, 则 $2^{16} + ab$ 就是 ab 的补码, $MUL(a,b)$ 在寄存器中就是 ab.

即公式 (1) 对于所有不溢出的情况是正确的.

若考虑溢出, 记最后的计算结果在寄存器中为 c, 则只能保证 $c \equiv ab (\bmod 2^{16})$, 参考下面的 C++ 程序运行结果, 可以认为是正确的.

```cpp
#include <iostream>

int main()
{
    int16_t num_pos_overflow = 500 * 433;
    int16_t num_neg_overflow = 500 * -433;
    std::cout << num_pos_overflow << std::endl;
    std::cout << num_neg_overflow << std::endl;
    return 0;
}
```

输出为, 其中 $500 \times 433 = 216500 \equiv 19892 (\bmod 2^{16})$

```
19892
-19892
```

1.2 L 版本程序

1.2.1 最初版本

根据公式 (1) 可以得到下面的算法:

Algorithm 1: Multiply

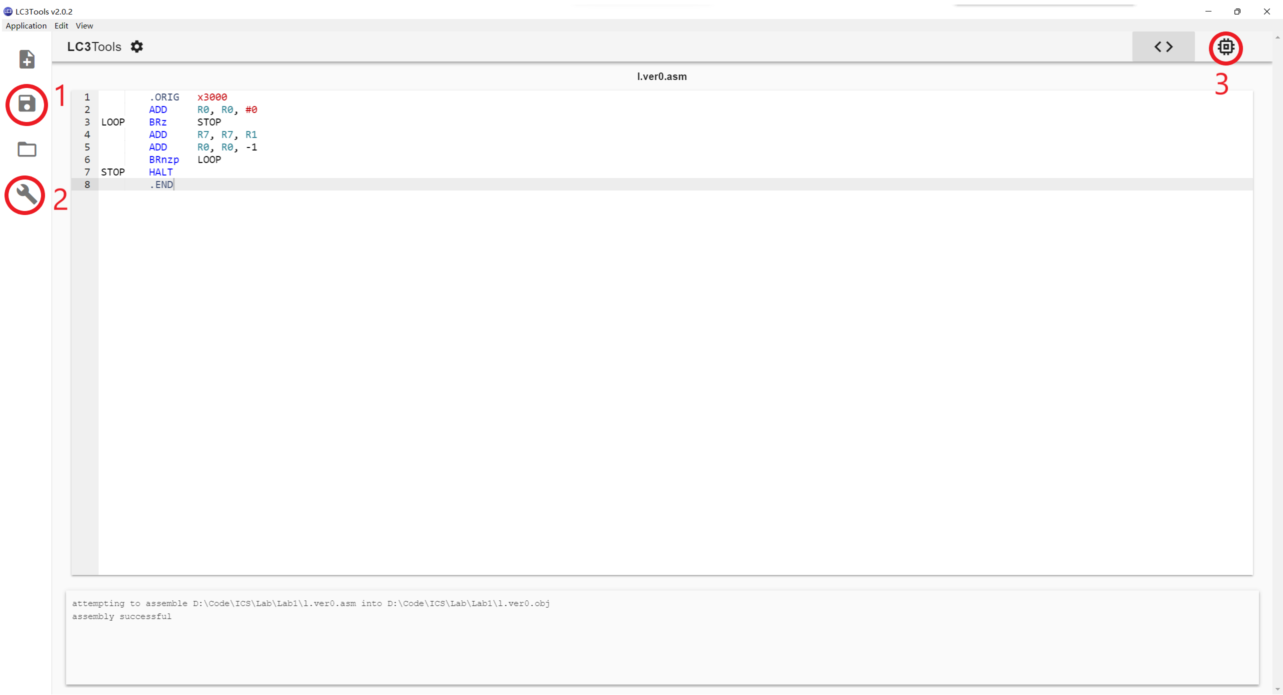
initial R2 to R7 $\leftarrow$ 0
loop:
if R0 = 0 **then**
  HALT
else
  R0 $\leftarrow$ R0 $-$ 1
  R7 $\leftarrow$ R7 + R1
  goto *loop*

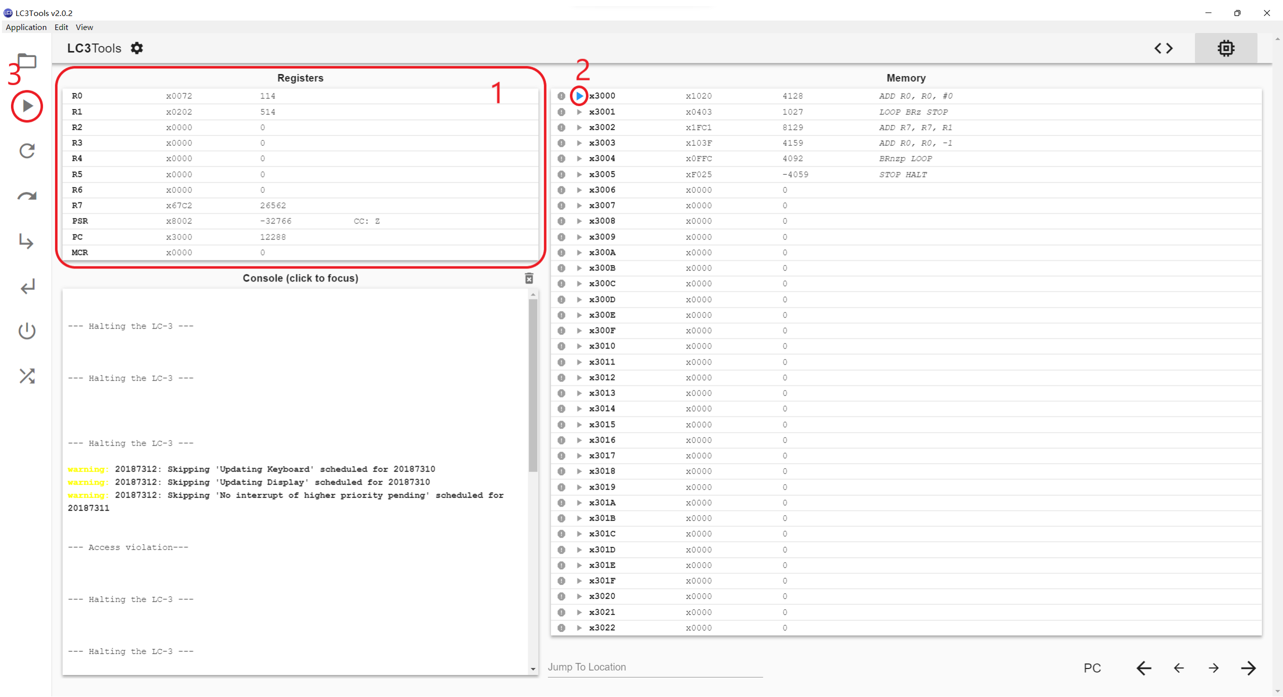
写成对应的汇编程序与机器码分别如下:

```
        ADD     R0, R0, #0
LOOP    BRz     STOP
        ADD     R7, R7, R1
        ADD     R0, R0, -1
        BRnzp   LOOP
STOP    HALT
```

```
0001000000100000
0000010000000011
0001111111000001
0001000000111111
0000111111111100
1111000000100101
```

程序总共 6 行, 使用 LC3Tools v2.0.2 进行测试, 在汇编程序的开头和结尾分别加上 `.ORIG x3000` 和 `.END`, 将 `.asm` 文件保存后, 点击左侧的 Assemble, 然后切换到模拟器页面, 在 Register 栏将寄存器设置为初始值后, 设置好 `PC`, 点击左侧的 Run 即可运行程序. 如下图





对测试样例进行测试, 结果如下

R0	R1	R7	R0×R1	理论值
1	1	1	1	1
5	4000	20000	20000	20000
4000	5	20000	20000	20000
-500	433	-19892	-216500	-19892
-114	-233	26562	26562	26562

可以看到, 程序运行没有问题.

1.2.2 优化与最终版本

考虑 $R1 = 0$ 的情况, 这时 R1 需要减去 2^{16} 次 1 才能重新为 0, 而 $2^{16}b \equiv 0 (\bmod 2^{16})$, 则如果没有第二行的 `BR` 指令, 并将第五行的 `BRnzp` 改为 `BRnp`, 程序仍然是正确的, 那么可以将汇编程序和机器码修改如下:

```
LOOP    ADD    R7, R7, R1
        ADD    R0, R0, -1
        BRnp   LOOP
STOP    HALT
```

```
0001111111000001
0001000000111111
0000101111111100
1111000000100101
```

可以看到程序从 6 行缩短到了 4 行, 按照上面的测试方法, 测试结果如下

R0	R1	R7	R0×R1	理论值
0	114	0	0	0
1	1	1	1	1
5	4000	20000	20000	20000
4000	5	20000	20000	20000
-500	433	-19892	-216500	-19892
-114	-233	26562	26562	26562

可以看到, 程序运行没有问题.

2 P 版本程序设计