

ICS Lab0 Report

虞佳焕
PB17121687

A. 算法设计

以 4 位有符号数为例说明，16 位有符号数只需要进行简单拓展。

设 4 位有符号数为 $A_3A_2A_1A_0$ ，右移后变为 $B_3B_2B_1B_0 = A_3A_3A_2A_1$ ，可以利用 AND 的掩码功能，设置一个只有第 i 位为 1 的原数掩码和一个只有第 $i-1$ 位为 1 的结果掩码。

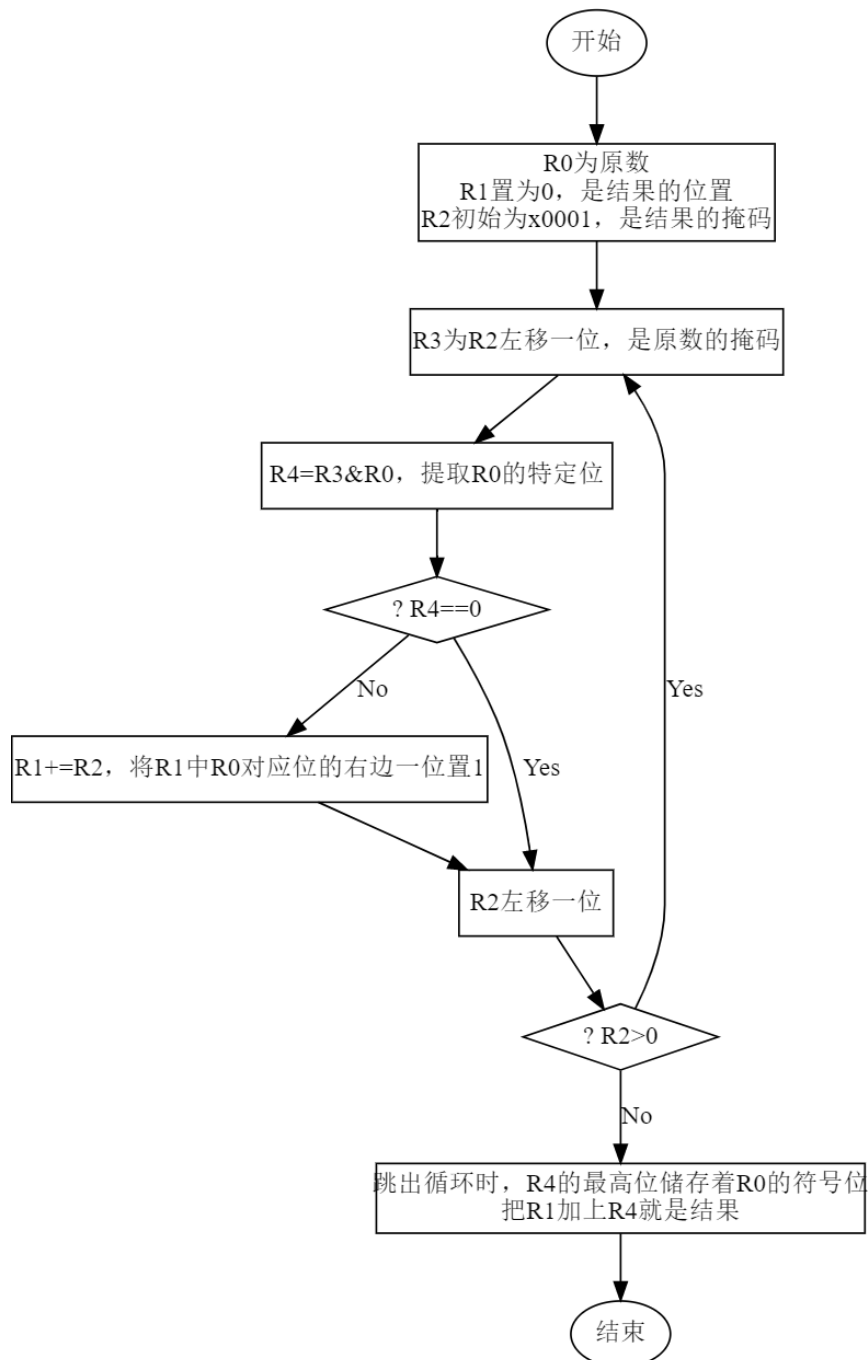
例如对于原数 $R0 = 1011$ ，原数掩码 $R3 = 0010$ ，结果 $R1$ 初始置为 0，结果掩码 $R2 = 0001$ 。计算 $R0 \& R3$ 即可判断 A_1 是否为 1，如果 A_1 为 1，则把 $R1$ 加上 $R2$ ，效果等价于把 B_0 置为与 A_1 相同。

在程序中把结果掩码 $R2$ 初始置为 1，每次循环都左移一位，直到为 1000 时跳出循环。

对于符号位 A_3 ，只需单独加上即可。

B. 编程

a) 根据以上算法，可绘制以下算法流程图：



b) 用 LC-3 汇编语言实现以上流程图：

AND R1, R1, #0 ;R1 置为 0, 是结果的储存位置

```

        ADD    R2, R1, #1    ;R2 初始置为 1, 是 R1 的掩码
        ;循环开始
LOOP    ADD    R3, R2, R2    ;R3 为 R2 左移一位, 是 R0 的掩码
        AND    R4, R0, R3    ;R4 为提取 R0 的特定位
        BRz    #1            ;R0 此位为 0 时不需要进行操作
        ADD    R1, R1, R2    ;R0 此位为 1 时, 用掩码将 R1 对应位的右边一位置为 1
        ADD    R2, R2, R2    ;R2 左移
        BRp    LOOP         ;R2 为 x8000 时跳出循环
        ;跳出循环时, R4 储存着 R0 的符号位
        ADD    R0, R1, R4    ;补上 R1 最左侧的符号位
        HALT                ;结束

```

c) 将汇编语言翻译成指令和机器码:

十六进制	二进制	指令
x5260	0101001001100000	AND R1, R1, #0
x1461	0001010001100001	ADD R2, R1, #1
x1682	0001011010000010	ADD R3, R2, R2
x5803	0101100000000011	AND R4, R0, R3
x0401	0000010000000001	BRz #1
x1242	0001001001000010	ADD R1, R1, R2
x1482	0001010010000010	ADD R2, R2, R2
x03FA	0000001111111010	BRp #-6
x1044	0001000001000100	ADD R0, R1, R4
xF025	1111000000100101	TRAP x25

d) 使用 hexedit 编写二进制文件 program.bin

由于需要将程序加载到 LC3 Simulate 的 x3000 位置开始执行, program.bin 的前两个字节应为 x3000, 使其符合 LC3 Simulator 的 obj 文件格式。

需要写入的内容为:

```

x3000
x5260
x1461
x1682
x5803
x0401
x1242
x1482
x03FA
x1044
xF025

```

以下是写入过程的部分截图:

```
monsoon@Monsoon-PC: /mnt/c/Users/rayomnd/Desktop
PS C:\Users\rayomnd\Desktop> bash
monsoon@Monsoon-PC: /mnt/c/Users/rayomnd/Desktop$ touch program.bin
monsoon@Monsoon-PC: /mnt/c/Users/rayomnd/Desktop$ hexedit program.bin
```

```
monsoon@Monsoon-PC: /mnt/c/Users/rayomnd/Desktop
0          30 00 52 60 14 61 16 82 58 03 04 01 12 42 14 82 03 FA 10 44 0.R`.a..X...B.....D
00000014   F0 25 _ .%
00000028
0000003C
00000050
00000064
00000078
0000008C
000000A0
000000B4
000000C8
000000DC
000000F0
00000104
00000118
0000012C
00000140
00000154
00000168
0000017C
00000190
000001A4
000001B8
000001CC
000001E0
000001F4
00000208
0000021C
00000230
00000244
00000258
0000026C
00000280
00000294
--**  program.bin      --0x16/0x16
```

```
monsoon@Monsoon-PC: /mnt/c/Users/rayomnd/Desktop
monsoon@Monsoon-PC:/mnt/c/Users/rayomnd/Desktop$ hexdump program.bin
00000000 0030 6052 6114 8216 0358 0104 4212 8214
00000010 fa03 4410 25f0
00000016
monsoon@Monsoon-PC:/mnt/c/Users/rayomnd/Desktop$
```

生成的文件能成功加载到 LC3 Simulate。

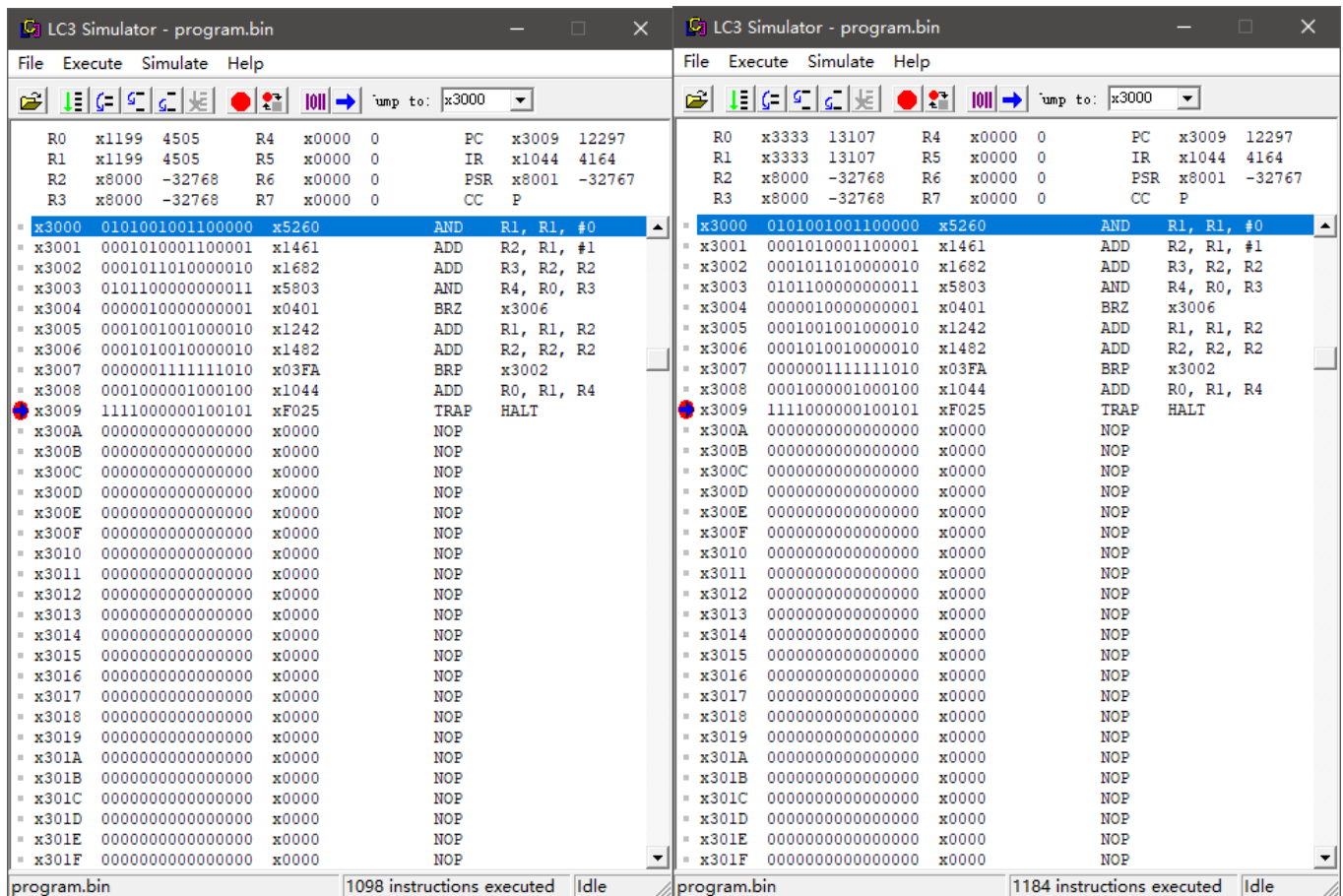
LC3 Simulator - program.bin									
File Execute Simulate Help									
R0	x048E	1166	R4	x0000	0	PC	x3000	12288	
R1	x048E	1166	R5	x0000	0	IR	x1044	4164	
R2	x8000	-32768	R6	x0000	0	PSR	x8001	-32767	
R3	x8000	-32768	R7	x0000	0	CC	P		
→ x3000	0101001001100000	x5260	AND	R1, R1, #0					
x3001	0001010001100001	x1461	ADD	R2, R1, #1					
x3002	0001011010000010	x1682	ADD	R3, R2, R2					
x3003	0101100000000011	x5803	AND	R4, R0, R3					
x3004	0000010000000001	x0401	BRZ	x3006					
x3005	0001001001000010	x1242	ADD	R1, R1, R2					
x3006	0001010010000010	x1482	ADD	R2, R2, R2					
x3007	0000001111111010	x03FA	BRP	x3002					
x3008	0001000001000100	x1044	ADD	R0, R1, R4					
x3009	1111000000100101	xF025	TRAP	HALT					
x300A	0000000000000000	x0000	NOP						
x300B	0000000000000000	x0000	NOP						
x300C	0000000000000000	x0000	NOP						
x300D	0000000000000000	x0000	NOP						
x300E	0000000000000000	x0000	NOP						
x300F	0000000000000000	x0000	NOP						
x3010	0000000000000000	x0000	NOP						
x3011	0000000000000000	x0000	NOP						
x3012	0000000000000000	x0000	NOP						
x3013	0000000000000000	x0000	NOP						
x3014	0000000000000000	x0000	NOP						
x3015	0000000000000000	x0000	NOP						
x3016	0000000000000000	x0000	NOP						
x3017	0000000000000000	x0000	NOP						
x3018	0000000000000000	x0000	NOP						
x3019	0000000000000000	x0000	NOP						
x301A	0000000000000000	x0000	NOP						
x301B	0000000000000000	x0000	NOP						
x301C	0000000000000000	x0000	NOP						
x301D	0000000000000000	x0000	NOP						
x301E	0000000000000000	x0000	NOP						
x301F	0000000000000000	x0000	NOP						
program.bin					601 instructions executed		Idle		

C. 测试

a) 个例测试

选择一些具有代表性的随机值和一些边界值进行个例测试。选择的量中应包含 0、边界值、正数和负数。

分别设置 R0 为 x0000, x8A9C, x7FFF, x8000, x2333, x6666 进行测试, 结果应该分别为 x0000, xC54E, x3FFF, xC000, x1199, x3333。



个例测试结果全部符合预期。

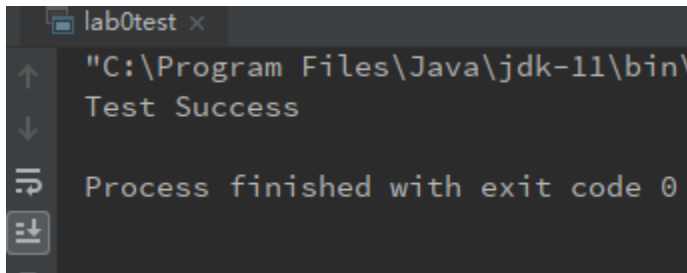
b) 自动测试

使用 Java 语言编写程序模拟指令的执行，以下为测试代码：

```
public class Lab0Test {
    public static void main(String args[]) {
        short[] R = new short[8]; //8 个寄存器
        for (int i = Short.MIN_VALUE; i <= Short.MAX_VALUE; i++) {
            R[0] = (short) i;
            //被测试代码开始
            R[1] = (short) (R[1] & 0); //AND R1, R1, #0
            R[2] = (short) (R[1] + 1); //ADD R2, R1, #1
            do {
                R[3] = (short) (R[2] + R[2]); //ADD R3, R2, R2
                R[4] = (short) (R[0] & R[3]); //AND R4, R0, R3
                if (R[4] != 0) //BRz #1
                    R[1] = (short) (R[1] + R[2]); //ADD R1, R1, R2
                R[2] = (short) (R[2] + R[2]); //ADD R2, R2, R2
            } while (R[2] > 0); //BRp LOOP
            R[0] = (short) (R[1] + R[4]); //ADD R0, R1, R4
            //被测试代码结束
            if (R[0] != (i >> 1)) {
                System.out.println("Test Failed");
                return;
            }
        }
    }
}
```

```
    System.out.println("Test Success");  
}  
}
```

运行结果如下：



```
lab0test x  
"C:\Program Files\Java\jdk-11\bin\  
Test Success  
Process finished with exit code 0
```

程序执行结果符合预期。