LAB02

——二进制除法

一. 实验要求

用十六进制编辑器(如 xxd、wxHexEditor、010Editor等),以LC-3 机器语言编写一个带有的程序,以便对给定值执行 1 位算术右移。

具体要求:

- •在 RO 寄存器中给出 16 位有符号整数输入值。输出值也要放置在 RO 寄存器中
 - •您的程序将从 x3000 加载并执行。
 - •执行的最后一条指令应该是 HALT。
 - •执行后, R7 寄存器应保持不变。

二.设计思路

RO:存放输入的数据

R1:存放与 R0 进行 AND 运算的操作数

R2:存放待加数

R3:存放计算结果;程序结束前将 R3 的值转入 R0 中

R4:计数器

R5:存放 AND 运算结果

R6:用于计算 x8000

利用"按位与"AND运算,依次判断R0中补码整数<mark>第2~16位</mark>是"0"还是"1",据此确定右移后结果<mark>第1~15位</mark>上是 "0"还是"1"

之后根据 R0 中数值的正负,来判断结果第 16 位补 "0" 还是 "1".

算法描述如下:

- (1) R1=x0002, R2=x0001, R3=x0000 R4=x000E, R6=x0000(初始化各寄存器)
- (2) R5=R1 & R0; 若 R5 等于 0, 转(4)
- (3) 将 R2 内容加到 R3
- (4) R2=R2<<1 , R1=R1<<1 (将 R2 与 R1 中的值都左移一位,以对 R0 下一位进行判断)
- (5) R4=R4-1; 若 R4>0,转 (2); 否则继续进行 (6)
- (6) 若 R0 非负,直接进行(7); 否则, R3=R3+x8000, 之后再进行(7)
- (7) 将 R3 中内容转移至 R0, 完毕。

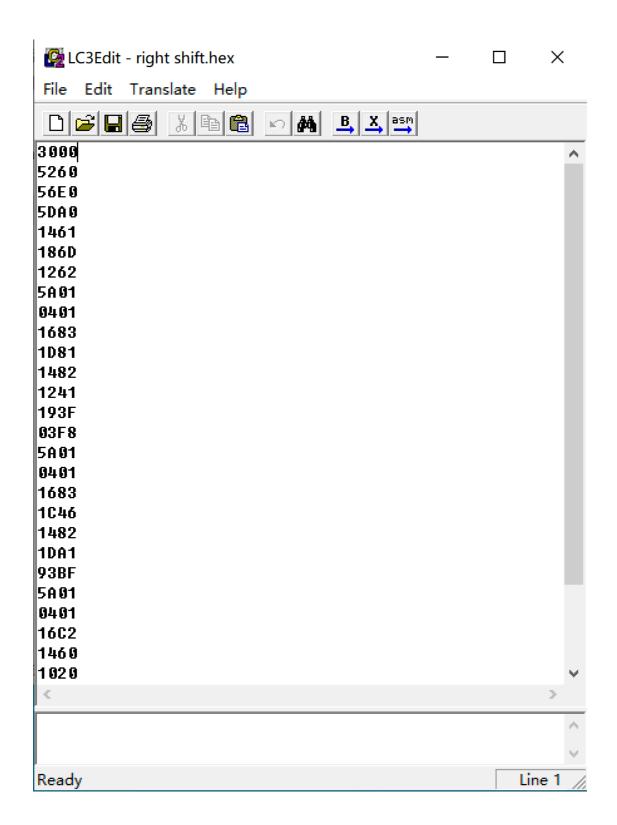
三. 编写过程

其中加法 (ADD)、按位与 (AND)、跳转 (BR) 由相应指令 来完成

左移<<运算用加法操作来完成; 当需要将 x4000 左移时, 由于 16 位补码加法产生溢出,故直接用赋值 x8000 来完成。

由于程序要求以 HALT 结尾,且 x8000 直接作为指令时非法,故无法从内存中直接 .FILL 装载 x8000; 我选择利用 R6 寄存器, 在前边的过程中, 先顺带将 R6 内容装为 x7FFF, 再 NOT 按位取反后得到 x8000, 之后便可以使用。

之后按照前面已经设计的算法,根据各步骤指令对应的机器码格式,即可写出对应的16进制机器码



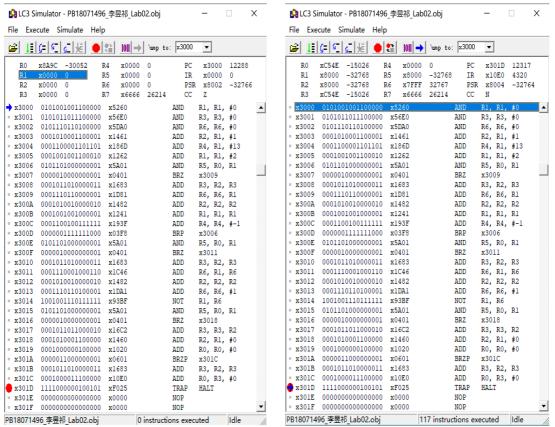
四. 测试用例

之后对生成的 .obj 文件进行测试,测试用例包含了负数(检验最高位是否可以补 "1")、奇数(检验最低位是否被丢弃)等

并且在测试前在 R7 中放置一个随机数,检验在程序执行到 HALT 前 R7 中内容是否会被改变

如下为使用讲义中给出的例子进行测试。程序执行前后 LC-3 模拟器状态如下:

(开始时 RO 装 x8A9C, R7 装 x6666)



可见与讲义中给出的结果(R0 装 xC54E、R7 仍为 x6666)一致之后又对其他数据进行了测试。

如下对 xCCCC 进行右移测试。测试前各 GPR 状态如下:

```
xCCCC -13108
                                 x0000
                                                        x3000
                                                               12288
    R1
         x0000
                                x0000
                                        n
                                                  IR
                                                        x0000
         x0000
                                x0000
                                        0
                                                       x8002
                                                               -32766
         x0000
                           R7
                                XFFFF
                                        -1
测试后:
         xE666 -6554
    R0
                           R4
                                x0000
                                                  PC
                                                        x301D
                                                               12317
    R1
         x8000
               -32768
                           R5
                                x8000
                                        -32768
                                                  IR
                                                        x10E0
                                                               4320
    R2
         x8000
                -32768
                           R6
                                        32767
                                                  PSR
                                                      x8004
                                x7FFF
                                                               -32764
    R3
         xE666
                -6554
                           R7
                                XFFFF
                                                  CC
```

可见 R0 中已经装入 xCCCC 右移一位的结果且最高位补 "1" 且 R7 中的数据在程序执行后并未发生改变

再对一些正数进行实验:

	R0	x0CFD	3325	R4	x0000	0	PC	x3000	12288
1	R1	x0000	0	R5	x0000	0	IR	x0000	0
1	R2	x0000	0	R6	x0000	0	PSR	x8002	-32766
	R3	x0000	0	R7	x0001	1	CC	Z	

执行程序后结果为:

R0	x067E	1662	R4	x0000	0	PC	x301D	12317	
R1	x8000	-32768	R5	x0000	0	IR	x10E0	4320	
R2	x8000	-32768	R6	x7FFF	32767	PSR	x8001	-32767	
R3	x067E	1662	R7	x0001	1	CC	P		

可见此时最高位会补 "0" 且最低位的 "1" 被丢弃。

之后还测试了其它用例,均成功得出正确的结果,在此不再一一列举。

五. 附录

"PB18071496_李昱祁_Lab02.pdf"

"PB18071496_李昱祁_Lab02.bin"