ICS Lab1 实验报告

张艺耀 PB20111630

L版本程序设计

最初版本：

```
0011 0000 0000 0000 ;start the program at x3000
;0101 111 111 1 00000    ;clear R7
0001 111 111 0 00 001   ;add R1 to R7, put result in R7  ;x3000
0001 000 000 1 00000    ;judge R0   ;x3001
0000 001 000000001  ;branch to location x3004 if  positive ;x3002
0001 000 000 1 00001    ;add 1 from R0,put result in R0    ;x3003
0000 100 000000001  ;branch to location x3006 if negative   ;x3004
0001 000 000 1 11111    ;sub 1 from R0, put result in R0    ;x3005
0000 101 111111001  ;branch to location x3000 if not zero    ;x3006
1111 0000 00100101  ;halt
```

净代码行数：7行。

思路是判断R0内值正负并不断执行加或减的操作，每次循环加一个R内值，结果存储在R7。

简化后版本：

```
0011 0000 0000 0000 ;start the program at x3000
0001 111 111 0 00 001   ;add R1 to R7,put result to R7
0001 000 000 1 11111    ;subtract 1 from R0, put result in R0
0000 101 111111101  ;branch to x3000 if pos or neg
1111 0000 00100101  ;halt
```

净代码行数：**3行。**

若两乘数中R0所在值是正数，则很显然可以得到结果；若R0为负数，则存储在内存中的值为R0的补码，十进制数对应R0 + 2^{16}。

则R1 * (R0 + 2^{16}) = R1 * R0 + R1 * 2^{16}。由于计算都是模2^{16}意义下的，故有

$R1 * R0 + R1 * 2^{16} \equiv R1 * R0 (mod 2^{16})$

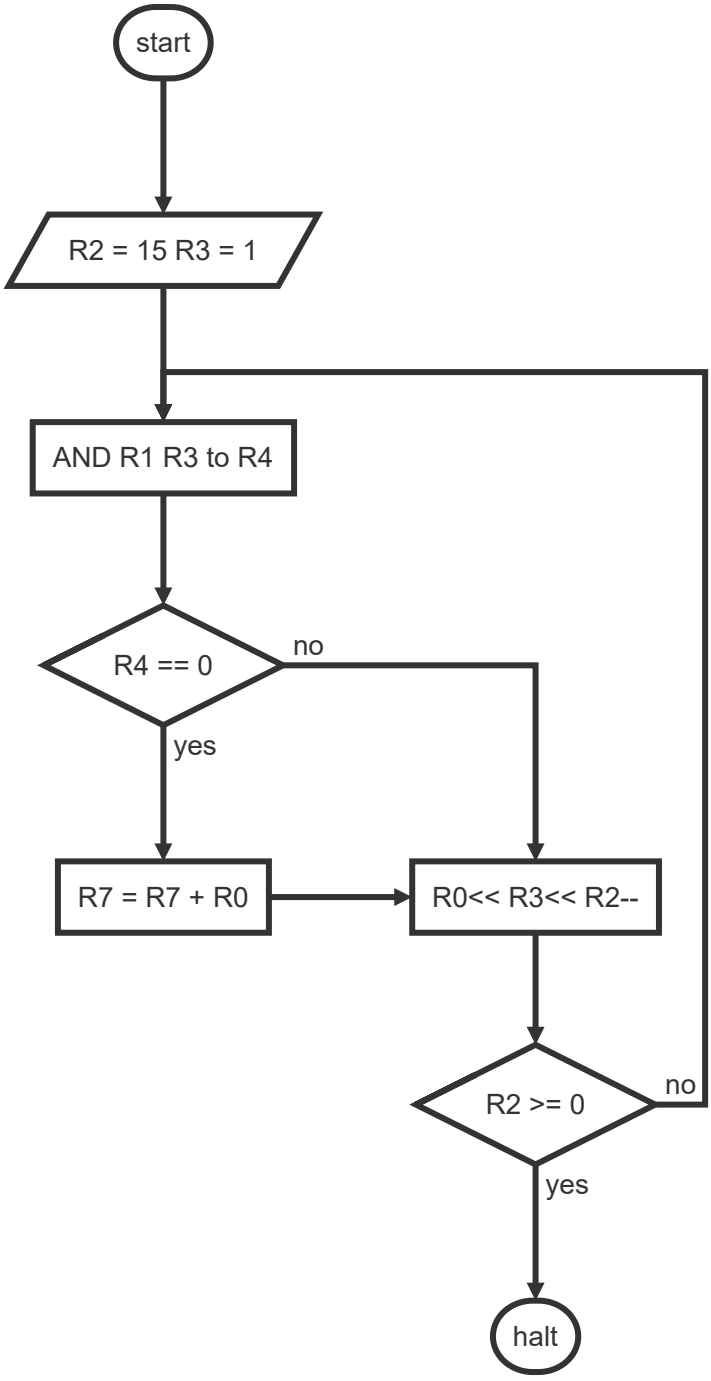
结果仍为R1*R0。

P版本程序设计

最初版本：

思路：R2用于计数；R3初始值为1，每次循环左移并与R1相与，用于判断R1相应位数值为0或1；R4用于记录R1当前位数是0或1；R7存储结果。

绘制流程图如下：



代码实现：

```
0011 0000 0000 0000 ;start from x3000
0001 010 010 1 01111    ;add 15 to R2
0001 011 011 1 00001    ;add 1 to R3

0101 100 001 0 00 011   ;and R1 R3 to R4
0000 010 000000001  ;if R4 == 0, PC++
0001 111 111 0 00 000   ;R7 = R7 + R0
0001 000 000 0 00 000   ;R0<<

0001 011 011 0 00 011   ;R3<<
0001 010 010 1 11111    ;R2 = R2 - 1
0000 011 111111001  ;if R2 >= 0, goto x3002
1111 0000 00100101  ;halt
```

每条循环执行7条指令，最多16次循环。初始指令有两条。故最多指令数：**114**条。

事实上，指令的多少取决于R1中0的多少，每个0使得每次循环执行6条语句；每个1使得每次循环执行7条语句。

优化版本：

可以将循环历程通过代码实现，使得指令条数减少到81条，代码如下：

```
0011 0000 0000 0000 ;start from x3000
0001 011 011 1 00001    ;add 1 to R3

0101 100 001 0 00 011   ;and R1 R3 to R4
0000 010 000000001  ;if R4 == 0, PC++
0001 111 111 0 00 000   ;R7 = R7 + R0
0001 000 000 0 00 000   ;R0<<
0001 011 011 0 00 011   ;R3<<
x16

1111 0000 00100101  ;halt
```

再优化：

前几次循环用立即数实现，将最后一次循环多余的移位操作删去：

```
0011 0000 0000 0000 ;start from x3000

0101 100 001 1 00001    ;and R1 00001 to R4
0000 010 000000001  ;if R4 == 0, PC++
0001 111 111 0 00 000   ;R7 = R7 + R0
0001 000 000 0 00 000   ;R0<<

0101 100 001 1 00010    ;and R1 00010 to R4
0000 010 000000001  ;if R4 == 0, PC++
0001 111 111 0 00 000   ;R7 = R7 + R0
0001 000 000 0 00 000   ;R0<<

0101 100 001 1 00100    ;and R1 00100 to R4
0000 010 000000001  ;if R4 == 0, PC++
0001 111 111 0 00 000   ;R7 = R7 + R0
0001 000 000 0 00 000   ;R0<<
```

```
0001 011 011 1 01000    ;add 1000 to R3

0101 100 001 0 00 011   ;and R1 R3 to R4
0000 010 000000001  ;if R4 == 0, PC++
0001 111 111 0 00 000   ;R7 = R7 + R0
0001 000 000 0 00 000   ;R0<<
0001 011 011 0 00 011   ;R3<<

0101 100 001 0 00 011   ;and R1 R3 to R4
0000 010 000000001  ;if R4 == 0, PC++
0001 111 111 0 00 000   ;R7 = R7 + R0
0001 000 000 0 00 000   ;R0<<
0001 011 011 0 00 011   ;R3<<

0101 100 001 0 00 011   ;and R1 R3 to R4
0000 010 000000001  ;if R4 == 0, PC++
0001 111 111 0 00 000   ;R7 = R7 + R0
0001 000 000 0 00 000   ;R0<<
0001 011 011 0 00 011   ;R3<<

0101 100 001 0 00 011   ;and R1 R3 to R4
0000 010 000000001  ;if R4 == 0, PC++
0001 111 111 0 00 000   ;R7 = R7 + R0
0001 000 000 0 00 000   ;R0<<
0001 011 011 0 00 011   ;R3<<

0101 100 001 0 00 011   ;and R1 R3 to R4
0000 010 000000001  ;if R4 == 0, PC++
0001 111 111 0 00 000   ;R7 = R7 + R0
0001 000 000 0 00 000   ;R0<<
0001 011 011 0 00 011   ;R3<<

0101 100 001 0 00 011   ;and R1 R3 to R4
0000 010 000000001  ;if R4 == 0, PC++
0001 111 111 0 00 000   ;R7 = R7 + R0
0001 000 000 0 00 000   ;R0<<
0001 011 011 0 00 011   ;R3<<

0101 100 001 0 00 011   ;and R1 R3 to R4
0000 010 000000001  ;if R4 == 0, PC++
0001 111 111 0 00 000   ;R7 = R7 + R0
0001 000 000 0 00 000   ;R0<<
0001 011 011 0 00 011   ;R3<<

0101 100 001 0 00 011   ;and R1 R3 to R4
0000 010 000000001  ;if R4 == 0, PC++
0001 111 111 0 00 000   ;R7 = R7 + R0
0001 000 000 0 00 000   ;R0<<
0001 011 011 0 00 011   ;R3<<

0101 100 001 0 00 011   ;and R1 R3 to R4
0000 010 000000001  ;if R4 == 0, PC++
0001 111 111 0 00 000   ;R7 = R7 + R0
0001 000 000 0 00 000   ;R0<<
0001 011 011 0 00 011   ;R3<<

0101 100 001 0 00 011   ;and R1 R3 to R4
0000 010 000000001  ;if R4 == 0, PC++
0001 111 111 0 00 000   ;R7 = R7 + R0
0001 000 000 0 00 000   ;R0<<
```

```
0001 011 011 0 00 011   ;R3<<

0101 100 001 0 00 011   ;and R1 R3 to R4
0000 010 000000001  ;if R4 == 0, PC++
0001 111 111 0 00 000   ;R7 = R7 + R0
0001 000 000 0 00 000   ;R0<<
0001 011 011 0 00 011   ;R3<<

0101 100 001 0 00 011   ;and R1 R3 to R4
0000 010 000000001  ;if R4 == 0, PC++
0001 111 111 0 00 000   ;R7 = R7 + R0
0001 000 000 0 00 000   ;R0<<
0001 011 011 0 00 011   ;R3<<

0101 100 001 0 00 011   ;and R1 R3 to R4
0000 010 000000001  ;if R4 == 0, PC++
0001 111 111 0 00 000   ;R7 = R7 + R0

1111 0000 00100101  ;halt
```

最差情况下指令条数：3 * 4 + 1 + 12 * 5 + 3 = 76条。

最好情况下指令条数：3 * 4 + 1 + 12 * 4 + 3 = 54条。

平均：65条指令。