

## 1. 实验题目: GREATEST COMMON DIVISOR

## 2. 实验要求

Your job:

Write a program in LC-3 assembly language and assemble it into LC-3 object file using an assembler.

The program is used to calculate the greatest common divisor of two positive numbers.

Details:

- Two positive 16-bit signed integers will be given in R0 and R1 register. And the output value should be put in R0.
- DO NOT access any part of memory other than x3000-xFDFF.
- The program should be ended with HALT.
- R7 register should remain the same after the execution

## 3. 实验思路&&代码讲解

### 三种写法的探讨:

法一考虑了储存和从高到低比较, 和我写的第二版本来说, 还是耗费了更多指令, 但我觉得这种储存思维对于要减的位数更多的测试情况会更好, 思路也比较特别, 特此记录 (内容很长因为是我第一版本的实验报告(^\_^))。

法二是我选择的“傻瓜”算法, 减数左移到最高位, 从被减数减去后, 重新循环, 左移到最高位, 再减去。根据我粗略地平均随机测试集指令数, 指令数与法三“查表法”相差不太多 (由于LC3位数少, 查表法的优化有限), 但非常节约空间, 也不需要做太多预先处理, 我个人认为这个算法更好。

法三是qq群里有人提到的“查表法”, 我没有具体写在报告中, 我认为内存空间同样需要节省。(图来自罗丽薇同学)

Performing the GCD algorithm recursively.

1. Both A and B are even,  $GCD(A, B) = 2 * GCD(A/2, B/2)$ ;
2. An odd and an even number, assuming that A is odd and B is even,  $GCD(A, B) = GCD(A, B/2)$ ;
3. Both A and B are odd,  $GCD(A, B) = GCD(abs(A-B)/2, min(A, B))$ . When  $A-B=0$ ,  $GCD=A$ ;

To perform  $/2$  efficiently, create an array Div2 before calculation. Furthermore,  $Div2[x] = x \gg 1$ .

### 法一:

关键思想:

利用欧几里得辗转相除法: 计算公式  $gcd(a, b) = gcd(b, a \bmod b)$

LC3并没有除法, 那就考虑减法。

考虑二进制形式, 存储减数的所有  $2^n$  次倍, 并从高到低逐一比较, 小于被减数就从被减数中减去相应的值, 减到没得减, 就交换被减数与减数。判断结束的条件是被减数为0。

具体思路:

初判断, 利用R2: R1>R0跳转函数3; R1=R0, HALT; <则顺序执行  
SPACE申请16个连续地址存减数, R5LEA存SPACE的第一个地址

函数0: 初始化计数器R3=0, R1取反加一的结果放进R4, 顺序执行

函数1: R4存进地址, R5地址++, 计数器R3++, R4左移, 是n就跳转函数1, 存完往下执行

函数2:

计数器--, BRn 函数3, 地址--, R2取出地址的内容, R2加R0

z: 跳转函数4

n: 跳回函数2

p: R0<-R2, 跳回函数2

函数3: 交换R0和R1, 利用R2, 结束后跳回函数0

函数4: R0<-R1, HALT

HALT

——对应代码讲解:

- 最初比较R1和R0的大小, 根据三种情况选择接下来的执行情况:

```
;first, specify R1 and R0
NOT R4,R1
ADD R4,R4,#1
ADD R2,R4,R0
BRZ THEEND
BRn F3

THEEND HALT
```

- 关于申请的16个内存空间

```
;save the address of the SPACE in R5
LEA R5,SPACE

SPACE .BLKW 16
```

- 函数0: 初始化计数器R3, R4=-R1

```
F0 AND R3,R3,#0
;R4 = -R1 in order to save the minus
NOT R4,R1
ADD R4,R4,#1
```

- 函数1: R4存入内存, R5指针++, 计数器++, R4左移, 若还是负数, 跳转函数1

```
F1 STR R4,R5,#0
ADD R5,R5,#1
ADD R3,R3,#1
ADD R4,R4,R4
BRn F1
```

- 函数2:

计数器--, 是负数跳转函数3, 否则地址--, R2取出地址的内容, R2+=R0

z: 跳转函数4

n: 跳回函数2

p:R0<-R2,跳回函数2

```
F2  ADD R3,R3,#-1
    BRn F3
    ADD R5,R5,#-1
    LDR R2,R5,#0
    ADD R2,R2,R0
    BRZ F4
    BRn F2
    AND R0,R0,#0
    ADD R0,R0,R2
    BR F2
```

- 函数3: 交换R0和R1, 结束后跳回函数0

```
F3  AND R2,R2,#0
    ADD R2,R2,R0 ;R2=R0
    AND R0,R0,#0
    ADD R0,R0,R1 ;R0=R1
    AND R1,R1,#0
    ADD R1,R1,R2 ;R1=R0
    BR F0
```

- 函数4: R0=R1, HALT

```
F4  AND R0,R0,#1
    ADD R0,R1,#0

THEEND  HALT
```

#### 4. 运行结果

本次实验指令数与测试数据关系很大, 我随机试了几个:

R0=#40, R1=#9102 813条指令

LC3 Simulator - try.obj

File Execute Simulate Help

Jump to: x3021

R0	x0002	2	R4	x0000	0	PC	x3021	12321
R1	x0002	2	R5	x0001	1	IR	x1060	4192
R2	x0000	0	R6	x0000	0	PSR	x8001	-32767
R3	x0001	1	R7	x0000	0	CC	P	

x3021	1111000000100101	xF025	THEEND	TRAP	HALT
x3022	0000000000000000	x0000	SPACE	NOP	
x3023	0000000000000000	x0000		NOP	
x3024	0000000000000000	x0000		NOP	
x3025	0000000000000000	x0000		NOP	
x3026	0000000000000000	x0000		NOP	
x3027	0000000000000000	x0000		NOP	
x3028	0000000000000000	x0000		NOP	
x3029	0000000000000000	x0000		NOP	
x302A	0000000000000000	x0000		NOP	
x302B	0000000000000000	x0000		NOP	
x302C	0000000000000000	x0000		NOP	
x302D	0000000000000000	x0000		NOP	
x302E	0000000000000000	x0000		NOP	
x302F	0000000000000000	x0000		NOP	
x3030	0000000000000000	x0000		NOP	
x3031	0000000000000000	x0000		NOP	
x3032	0000000000000000	x0000		NOP	
x3033	0000000000000000	x0000		NOP	
x3034	0000000000000000	x0000		NOP	
x3035	0000000000000000	x0000		NOP	
x3036	0000000000000000	x0000		NOP	
x3037	0000000000000000	x0000		NOP	
x3038	0000000000000000	x0000		NOP	
x3039	0000000000000000	x0000		NOP	
x303A	0000000000000000	x0000		NOP	
x303B	0000000000000000	x0000		NOP	
x303C	0000000000000000	x0000		NOP	

try.obj 813 instructions executed Idle

R0=#6, R1=#8 370条指令

LC3 Simulator - try.obj

File Execute Simulate Help

Jump to: x3021

R0	x0002	2	R4	x0000	0	PC	x3021	12321
R1	x0002	2	R5	x0000	0	IR	x1060	4192
R2	x0000	0	R6	x0000	0	PSR	x8001	-32767
R3	x0000	0	R7	x0000	0	CC	P	









x3021	1111000000100101	xF025	THEEND	TRAP	HALT
x3022	0000000000000000	x0000	SPACE	NOP	
x3023	0000000000000000	x0000		NOP	
x3024	0000000000000000	x0000		NOP	
x3025	0000000000000000	x0000		NOP	
x3026	0000000000000000	x0000		NOP	
x3027	0000000000000000	x0000		NOP	
x3028	0000000000000000	x0000		NOP	
x3029	0000000000000000	x0000		NOP	
x302A	0000000000000000	x0000		NOP	
x302B	0000000000000000	x0000		NOP	
x302C	0000000000000000	x0000		NOP	
x302D	0000000000000000	x0000		NOP	
x302E	0000000000000000	x0000		NOP	
x302F	0000000000000000	x0000		NOP	
x3030	0000000000000000	x0000		NOP	
x3031	0000000000000000	x0000		NOP	
x3032	0000000000000000	x0000		NOP	
x3033	0000000000000000	x0000		NOP	
x3034	0000000000000000	x0000		NOP	
x3035	0000000000000000	x0000		NOP	
x3036	0000000000000000	x0000		NOP	
x3037	0000000000000000	x0000		NOP	
x3038	0000000000000000	x0000		NOP	
x3039	0000000000000000	x0000		NOP	
x303A	0000000000000000	x0000		NOP	
x303B	0000000000000000	x0000		NOP	
x303C	0000000000000000	x0000		NOP	

try.obj 370 instructions executed Idle

R0=#11, R1=#1100 152条指令

LC3 Simulator - try.obj

File Execute Simulate Help

 Jump to:

R0	x000B	11	R4	x5000	20480	PC	x3021	12321
R1	x000B	11	R5	x0002	2	IR	x1060	4192
R2	x0000	0	R6	x0000	0	PSR	x8001	-32767
R3	x0002	2	R7	x0000	0	CC	P	

x3000	1001100001111111	x987F	NOT	R4, R1
x3001	0001100100100001	x1921	ADD	R4, R4, #1
x3002	0001010100000000	x1500	ADD	R2, R4, R0
x3003	00000100000011101	x041D	BRZ	THEEND
x3004	0000100000010011	x0813	BRN	F3
x3005	1110101000011100	xEA1C	LEA	R5, SPACE
x3006	0101011011100000	x56E0	F0 AND	R3, R3, #0
x3007	1001100001111111	x987F	NOT	R4, R1
x3008	0001100100100001	x1921	ADD	R4, R4, #1
x3009	0111100101000000	x7940	F1 STR	R4, R5, #0
x300A	0001101101100001	x1B61	ADD	R5, R5, #1
x300B	0001011011100001	x16E1	ADD	R3, R3, #1
x300C	0001100100000100	x1904	ADD	R4, R4, R4
x300D	0000100111111011	x09FB	BRN	F1
x300E	0001011011111111	x16FF	F2 ADD	R3, R3, #-1
x300F	0000100000001000	x0808	BRN	F3
x3010	0001101101111111	x1B7F	ADD	R5, R5, #-1
x3011	0110010101000000	x6540	LDR	R2, R5, #0
x3012	0001010010000000	x1480	ADD	R2, R2, R0
x3013	0000010000001011	x040B	BRZ	F4
x3014	0000100111111001	x09F9	BRN	F2
x3015	0101000000100000	x5020	AND	R0, R0, #0
x3016	0001000000000010	x1002	ADD	R0, R0, R2
x3017	0000111111110110	x0FF6	BRNZP	F2
x3018	0101010010100000	x54A0	F3 AND	R2, R2, #0
x3019	0001010010000000	x1480	ADD	R2, R2, R0
x301A	0101000000100000	x5020	AND	R0, R0, #0
x301B	0001000000000001	x1001	ADD	R0, R0, R1

try.obj

152 instructions executed

Idle

## 5. 关于时空复杂度

——“Analyze the space and time complexity of your algorithm in your report.”

代码34个空间，数据相关占用16个空间，复杂度为O(1)。

时间最长的情况可以想象是一直在辗转减、减小得很慢，是对数级别的复杂度(log)。

## 6. 关于输入含负数或0

——“Think about handling negative and zero inputs in your report”

加入预判断：

如果有0，最大公因数为另一个数；

如果有负数，取反加一后开始本程序的运算，但有例外情况，即x8000 (-32768)，可以任意加上或减去另一个数，因为不影响辗转相除法的算法，之后重新进入预判断即可。

## 7. 源代码

```
.ORIG    x3000
;first, specify R1 and R0
NOT     R4,R1
```

```

ADD R4,R4,#1
ADD R2,R4,R0
BRZ THEEND
BRn F3
;save the address of the SPACE in R5
LEA R5,SPACE

F0  AND R3,R3,#0
    ;R4 = -R1 in order to save the minus
    NOT R4,R1
    ADD R4,R4,#1
F1  STR R4,R5,#0
    ADD R5,R5,#1
    ADD R3,R3,#1
    ADD R4,R4,R4
    BRn F1
F2  ADD R3,R3,#-1
    BRn F3
    ADD R5,R5,#-1
    LDR R2,R5,#0
    ADD R2,R2,R0
    BRZ F4
    BRn F2
    AND R0,R0,#0
    ADD R0,R0,R2
    BR F2
F3  AND R2,R2,#0
    ADD R2,R2,R0 ;R2=R0
    AND R0,R0,#0
    ADD R0,R0,R1 ;R0=R1
    AND R1,R1,#0
    ADD R1,R1,R2 ;R1=R0
    BR F0
F4  AND R0,R0,#1
    ADD R0,R1,#0
THEEND HALT
SPACE .BLKW 16

.END

```

法二：

#### 【关键思想】

利用欧几里得辗转相除法：计算公式 $\gcd(a,b) = \gcd(b, a \bmod b)$

LC3并没有除法，那就考虑减法。减数左移到最高位，从被减数减去后，重新循环，左移到最高位，再减去。（模仿竖式除法）

#### 【代码】

```

.ORIG x3000
;函数0: 利用R2: R1>R0跳转函数3; R1=R0, HALT; <则顺序执行
F0  NOT R2,R0
    ADD R2,R2,#1
    ADD R2,R2,R1
    BRp F3
    BRZ END

```



;函数1: 初始化R3=-R1,R4=R3(=-R1)

```
F1 NOT R3,R1
    ADD R3,R3,#1
    ADD R4,R3,#0
```

;函数2: R4\*2, R5=R0-R4,

```
; p LEFT
; z FINAL
```

; n R0--R3,跳回函数0

```
F2 ADD R4,R4,R4
    ADD R5,R0,R4
    BRp LEFT
    BRZ FINAL
    ADD R0,R0,R3
    BR F0
```

;LEFT: R3\*2, BR 函数2

```
LEFT ADD R3,R3,R3
      BR F2
```

;函数3: 交换R0和R1,利用R2, 结束后跳回函数1

```
F3 ADD R2,R0,#0
    ADD R0,R1,#0
    ADD R1,R2,#0
    BR F1
```

;FINAL:R0=R1+#0

;HALT

```
FINAL ADD R0,R1,#0
```

```
END HALT
```

```
.END
```

## 【测试】

1. 指令数最多的可能性: R1=x7fff, R0=x0001, 729条指令

LC3 Simulator - PB18000028_邱子悦_Lab032.obj									
File Execute Simulate Help									
[Icons] Dump to: x3000									
R0	x0001	1	R4	xFFFC	-4	PC	x3015	12309	
R1	x0001	1	R5	xFFFF	-1	IR	x0410	1040	
R2	x0000	0	R6	x0000	0	PSR	x8002	-32766	
R3	xFFFE	-2	R7	x0000	0	CC	Z		
x3000 1001010000011111 x943F F0 NOT R2, R0									
x3001 0001010010100001 x14A1 ADD R2, R2, #1									
x3002 0001010010000001 x1481 ADD R2, R2, R1									
x3003 0000001000001100 x020C BRP F3									
x3004 0000010000001000 x0410 BRZ END									
x3005 1001011001111111 x967F F1 NOT R3, R1									
x3006 0001011011100001 x16E1 ADD R3, R3, #1									
x3007 0001100011100000 x18E0 ADD R4, R3, #0									
x3008 0001100100000100 x1904 F2 ADD R4, R4, R4									
x3009 0001101000000100 x1A04 ADD R5, R0, R4									
x300A 0000001000000011 x0203 BRP LEFT									
x300B 0000010000001000 x0408 BRZ FINAL									
x300C 0001000000000011 x1003 ADD R0, R0, R3									
x300D 0000111111110010 x0FF2 BRNFP F0									
x300E 0001011011000011 x16C3 LEFT ADD R3, R3, R3									
x300F 0000111111111000 x0FF8 BRNFP F2									
x3010 0001010000010000 x1420 F3 ADD R2, R0, #0									
x3011 0001000001100000 x1060 ADD R0, R1, #0									
x3012 0001001010100000 x12A0 ADD R1, R2, #0									
x3013 0000111111110001 x0FF1 BRNFP F1									
x3014 0001000001100000 x1060 FINAL ADD R0, R1, #0									
x3015 1111000000100101 xF025 END TRAP HALT									
x3016 0000000000000000 x0000 NOP									
x3017 0000000000000000 x0000 NOP									
x3018 0000000000000000 x0000 NOP									
x3019 0000000000000000 x0000 NOP									
x301A 0000000000000000 x0000 NOP									
x301B 0000000000000000 x0000 NOP									
PB18000028_邱子悦_Lab032.obj					729 instructions executed			Idle	

2. 随机生成: R1=x3930, R0=x2820, 154条指令



LC3 Simulator - PB18000028\_邱子悦\_Lab032.obj

File Execute Simulate Help

Jump to: x3000

R0	x0030	48	R4	xFFA0	-96	PC	x3015	12309
R1	x0030	48	R5	x0000	0	IR	x1060	4192
R2	x0030	48	R6	x0000	0	PSR	x8001	-32767
R3	xFFD0	-48	R7	x0000	0	CC	P	

```

x3000 1001010000111111 x943F F0 NOT R2, R0
x3001 0001010010100001 x14A1 ADD R2, R2, #1
x3002 0001010010000001 x1481 ADD R2, R2, R1
x3003 0000001000001100 x020C BRP F3
x3004 0000010000010000 x0410 BRZ END
x3005 1001011001111111 x967F F1 NOT R3, R1
x3006 0001011011100001 x16E1 ADD R3, R3, #1
x3007 0001100011100000 x18E0 ADD R4, R3, #0
x3008 0001100100000100 x1904 F2 ADD R4, R4, R4
x3009 0001101000000100 x1A04 ADD R5, R0, R4
x300A 0000001000000011 x0203 BRP LEFT
x300B 0000010000001000 x0408 BRZ FINAL
x300C 0001000000000011 x1003 ADD R0, R0, R3
x300D 0000111111110010 x0FF2 BRN ZP F0
x300E 0001011011000011 x16C3 LEFT ADD R3, R3, R3
x300F 0000111111111000 x0FF8 BRN ZP F2
x3010 0001010000100000 x1420 F3 ADD R2, R0, #0
x3011 0001000001100000 x1060 ADD R0, R1, #0
x3012 0001001010100000 x12A0 ADD R1, R2, #0
x3013 0000111111110001 x0FF1 BRN ZP F1
x3014 0001000001100000 x1060 FINAL ADD R0, R1, #0
x3015 1111000000100101 xF025 END TRAP HALT
x3016 0000000000000000 x0000 NOP
x3017 0000000000000000 x0000 NOP
x3018 0000000000000000 x0000 NOP
x3019 0000000000000000 x0000 NOP
x301A 0000000000000000 x0000 NOP
x301B 0000000000000000 x0000 NOP
  
```

PB18000028\_邱子悦\_Lab032.obj 154 instructions executed Idle

3. 指令数比较少的情况: R0=x0008, R1=x0006, 41条指令

LC3 Simulator - PB18000028\_邱子悦\_Lab032.obj

File Execute Simulate Help

Jump to: x3000

R0	x0002	2	R4	xFF8	-8	PC	x3015	12309
R1 <td>x0002 <td>2 <td>R5 <td>xFFE <td>-2 <td>IR <td>x0410 <td>1040</td> </td></td></td></td></td></td></td>	x0002 <td>2 <td>R5 <td>xFFE <td>-2 <td>IR <td>x0410 <td>1040</td> </td></td></td></td></td></td>	2 <td>R5 <td>xFFE <td>-2 <td>IR <td>x0410 <td>1040</td> </td></td></td></td></td>	R5 <td>xFFE <td>-2 <td>IR <td>x0410 <td>1040</td> </td></td></td></td>	xFFE <td>-2 <td>IR <td>x0410 <td>1040</td> </td></td></td>	-2 <td>IR <td>x0410 <td>1040</td> </td></td>	IR <td>x0410 <td>1040</td> </td>	x0410 <td>1040</td>	1040
R2 <td>x0000</td> <td>0</td> <td>R6 <td>x0000</td> <td>0</td> <td>PSR <td>x8002</td> <td>-32766</td> </td></td>	x0000	0	R6 <td>x0000</td> <td>0</td> <td>PSR <td>x8002</td> <td>-32766</td> </td>	x0000	0	PSR <td>x8002</td> <td>-32766</td>	x8002	-32766
R3 <td>xFFC</td> <td>-4</td> <td>R7 <td>x0000</td> <td>0</td> <td>CC <td>Z</td> <td></td> </td></td>	xFFC	-4	R7 <td>x0000</td> <td>0</td> <td>CC <td>Z</td> <td></td> </td>	x0000	0	CC <td>Z</td> <td></td>	Z	

```

x3000 1001010000111111 x943F F0 NOT R2, R0
x3001 0001010010100001 x14A1 ADD R2, R2, #1
x3002 0001010010000001 x1481 ADD R2, R2, R1
x3003 0000001000001100 x020C BRP F3
x3004 0000010000010000 x0410 BRZ END
x3005 1001011001111111 x967F F1 NOT R3, R1
x3006 0001011011100001 x16E1 ADD R3, R3, #1
x3007 0001100011100000 x18E0 ADD R4, R3, #0
x3008 0001100100000100 x1904 F2 ADD R4, R4, R4
x3009 0001101000000100 x1A04 ADD R5, R0, R4
x300A 0000001000000011 x0203 BRP LEFT
x300B 0000010000001000 x0408 BRZ FINAL
x300C 0001000000000011 x1003 ADD R0, R0, R3
x300D 0000111111110010 x0FF2 BRN ZP F0
x300E 0001011011000011 x16C3 LEFT ADD R3, R3, R3
x300F 0000111111111000 x0FF8 BRN ZP F2
x3010 0001010000100000 x1420 F3 ADD R2, R0, #0
x3011 0001000001100000 x1060 ADD R0, R1, #0
x3012 0001001010100000 x12A0 ADD R1, R2, #0
x3013 0000111111110001 x0FF1 BRN ZP F1
x3014 0001000001100000 x1060 FINAL ADD R0, R1, #0
x3015 1111000000100101 xF025 END TRAP HALT
x3016 0000000000000000 x0000 NOP
x3017 0000000000000000 x0000 NOP
x3018 0000000000000000 x0000 NOP
x3019 0000000000000000 x0000 NOP
x301A 0000000000000000 x0000 NOP
x301B 0000000000000000 x0000 NOP
  
```

PB18000028\_邱子悦\_Lab032.obj 41 instructions executed Idle

### 【关于时空复杂度】

——“Analyze the space and time complexity of your algorithm in your report.”

代码24个空间, 复杂度为O(1)。

时间最长的情况可以想象是一直在辗转减、减小得很慢，是对数级别的复杂度(log)。

### 【关于输入含负数或0】

——“Think about handling negative and zero inputs in your report”

加入预判断：

如果有0，最大公因数为另一个数；

如果有负数，取反加一后开始本程序的运算，但有例外情况，即x8000 (-32768)，可以任意加上或减去另一个数，因为不影响辗转相除法的算法，之后重新进入预判断即可。