

LAB03

——PB18071496_李昱祁

一 . 实验要求 :

用 LC-3 汇编语言编写程序, 并用汇编程序将其汇编成 LC-3 .obj 文件。该程序用于计算两个正数的最大公约数。
具体 :

- 在 R0 和 R1 寄存器中将给出两个正数 16 位有符号整数。
(输出值应该放在 R0 中)
- 请勿访问除 x3000~xFDFF 以外的任何内存部分。
- 程序应以 HALT 结束。
- 执行后, R7 寄存器应保持不变。

二 . 算法实现

采用 “更相减损术” :

设两个数分别为 a、b, 则求 a、b 最大公约数算法如下 :

- (1) 计算 $a-b$
- (2) 若结果等于 0, 转 (3) ;
若结果大于 0, 则令 $a = a-b$, 转 (1) ;
否则, 令 $b=b-a$, 转 (1)
- (3) 此时 a 值即为所求最大公约数

三 . 程序编写

在 visual studio code 编辑器中使用 LC-3 汇编语言, 按照上述算法编写了程序, 源代码与注释如下 :

```
.ORIG x3000
loop    NOT R2,R1

    ADD R2,R2,#1

    ADD R2,R0,R2    ;calculate R0-R1

    BRN Lable      ;if R0<R1 , go to "Lable" (to assign R1 = R1 - R0);
```

```

BRZ end          ;if R1==R0 ,the integer in R0 is the
ADD R0,R2,#0     ;Neither of the above is true,which means R0>R1.
;then assign R0 = R0 - R1.

BRNZP loop
Lable NOT R2,R2

ADD R1,R2,#1     ;assign R1 = R1 - R0

BRNZP loop      ;Continue the circulation

end HALT

.END

```

其中当 $R0 - R1 < 0$ 时, $R1 - R0 == \sim(R0 - R1) + 1$, 可以使用之前运算的结果节省指令数

四 . 性能分析

1.时间复杂度：

设 $N = \max(a,b)$

(1) 若 a 、 b 两数大小比较接近, 则程序其运行情况接近于“辗转相除法” (此时相减与取模等效), 其时间复杂度平均约为 $O(\log N)$

(2) 最坏情况下为 $O(N)$. 即若 a 、 b 两数差距较大($a \gg b$ 或 $a \ll b$), 此时会重复多次执行某一种情况, 程序性能较低

综上, 该算法的时间复杂度在 $O(\log N) \sim O(N)$ 之间, 视两个源操作数大小差异是否很大而改变

2.空间复杂度：

因为本程序仅使用了三个通用寄存器 (包括 $R0$ 、 $R1$), 空间复杂度为 $O(1)$

3.使用的指令数：

使用高级语言程序生成五组, 每组 1000 个数据的随机数

(1 ~ 32767 之间, 等概率取值)

产生数据如下：

之后在 LC-3 编辑器中使用十六进制代码输入，在程序运行前先加载至内存从 x4000 开始的内存中；
 设置一个循环，每次读取数据入 R0,R1 中，计算它们的最大公因数，结束后读下一组数据
 结果如下：

LC3 Simulator - GREATEST COMMON DIVISOR.obj

File Execute Simulate Help

Jump to: x3000

Register	Value	Register	Value	Register	Value
R0	x0001 1	R4	x0000 0	PC	x3011 12305
R1	x0001 1	R5	x0000 0	IR	x040C 1036
R2	x0000 0	R6	x0000 0	PSR	x8002 -32766
R3	x43E7 17383	R7	x0000 0	CC	Z

Address	Hex	Binary	Label	Instruction
x3000	00100110000010001	x2611		LD R3, Address
x3001	00101000000010001	x2811		LD R4, counter
x3002	0001011011100001	x16E1	next	ADD R3, R3, #1
x3003	0001100100111111	x193F		ADD R4, R4, #-1
x3004	0000010000001100	x040C		BRZ end
x3005	0110001011000000	x62C0		LDR R1, R3, #0
x3006	0110000011000001	x60C1		LDR R0, R3, #1
x3007	1001010001111111	x947F	loop	NOT R2, R1
x3008	0001010010100001	x14A1		ADD R2, R2, #1
x3009	0001010000000010	x1402		ADD R2, R0, R2
x300A	0000100000000011	x0803		BRN Lable
x300B	0000010111110110	x05F6		BRZ next
x300C	0001000010100000	x10A0		ADD R0, R2, #0
x300D	0000111111111001	x0FF9		BRNZP loop
x300E	1001010010111111	x94BF	Lable	NOT R2, R2
x300F	0001001010100001	x12A1		ADD R1, R2, #1
x3010	0000111111110110	x0FF6		BRNZP loop
x3011	1111000000100101	xF025	end	TRAP HALT
x3012	0011111111111111	x3FFF	Address	ST R7, Address
x3013	0000001111101000	x03E8	counter	BRP x2FFC
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

GREATEST COMMON DIVISOR.obj | 488578 instructions executed | Idle

则平均每组数据所用的指令数为 $(488578 - 2) / 999 - 5 = 484$

(减去了循环读取数据时所用的指令条数)

另外两次测试中, 求得的平均指令数也在 500 条左右

五. 程序加速:

在上文分析该程序的时间复杂度时已经得出, 在两个数据相差较大时更相减损术效率较低, 可以采用 stein 算法:

(1): 当两数均为偶数时将其同时除以 2 至至少一数为奇数为止, 记录除掉的所有公因数 2 的乘积 k ;

(2): 如果仍有一数为偶数, 连续除以 2 直至该数为奇数为止;

(3): 用更相减损法(辗转相减法), 求出两奇数的最大公约数 d ;

(4): 原来两数的最大公约数即为 $d*k$;

判断奇偶性可以通过 &1 (AND 指令) 实现, 除以 2 可通过右移实现。

由于需要进行右移运算而 LC-3 计算机指令中无该指令, 为了提高程序运行速度, 我们先将从 $x0002$ - $x7FFE$ 的所有偶数的 $1/2$ 装载入 LC-3 的内存中 (例如从 $x5000$ 开始), 在需要进行右移运算时直接查表求得

这种算法可将平均的时间复杂度将至 $O(\log N)$ 左右, 但是空间复杂度将大幅度增加, 占用了 LC-3 一半的内存用于快速实现右移运算

六. 其它要求:

(处理 0 和负值输入)

根据定义, 最大公因数的概念是相对于自然数的, 所以 0 和负数显然为非法输入; 且若原始程序出现了 0, 还会造成死循环。

我们可以扩展定义:

(1) 0 和非负数的最大公因数为该非负数

(2) 若输入中出现负数, 则将该负数转变为相反数后再进行计算

在开始时对两个输入进行判断 (判断时可采用 $ADD R1, R1, \#0$ 及 BRn 、 BRz 进行跳转), 先判断 $R0$:

若 $R0 < 0$, 则 $R0 = -R0$;

若 $R0 = 0$ 则 $R0 = |R1|$;

若 $R0 > 0$ 且 $R1 = 0$, 直接结束 (此时 $R0$ 已经装载了最终的结果)

若 $R1 < 0$, 则 $R1 = -R1$, 之后再使用更相减损术计算

若 $R1 > 0$, 直接使用更相减损术计算

七．实验总结：

- (1) 回顾了求最大公约数的几种算法
- (2) 查表法是一种常见的程序优化方法，牺牲空间用量来换取时间上的性能
- (3) 通过查阅资料，了解了快速求解最大公因数在现代密码学中的应用