**0** addi $a0, $zero, 5 #$a0赋初值5

**1** xor $v0, $zero, $zero #$v0赋初值0

**2** jal sum #跳转到sum标签并将3号指令地址存入$ra中

**Loop:**

**3** beq $zero, $zero, Loop #死循环作为程序的结束

**sum:**

**4** addi $sp, $sp, -8 #开辟两条指令大小的栈空间

**5** sw $ra, 4($sp) #存入jal指令写入$ra的地址

**6** sw $a0, 0($sp) #存入$a0的值

**7** slti $t0, $a0, 1 #a0值每次减1，在a0减小到0时t0为1，此时下一条分支条件不成立

**8** beq $t0, $zero, L1 #结合上一条指令控制入栈次数，保证入栈的数据是n,n-1,n-2,……1

**9** addi $sp, $sp, 8 #栈指针上移两个数据位，这条指令只用一次，跳过a0等于零时存入的a0值和地址，因为加0不改变v0的值

**10** jr $ra #只使用一次，对正整数n来说a0减小到0一定是从L1跳回sum，因此ra存14号指令地址

**L1:**

**11** add $v0, $a0, $v0 #入栈过程累加a0(从n加到1)

**12** addi $a0, $a0, -1 #a0每次减1，用于下次累加

**13** jal sum #跳到sum，继续入栈

**14** lw $a0, 0($sp) # 取出栈里的a0

**15** lw $ra, 4($sp) #从栈里取出之前存的jal地址，出栈过程在14-18循环，栈顶存的不是14号指令地址，是最初的3号指令地址，之后进入死循环

**16** addi $sp, $sp, 8 #栈指针上移两个数据位

**17** add $v0, $a0, $v0 #继续累加a0，出栈过程从1加到n

**18** jr $ra #跳回14号指令，最后一次跳到3号指令

功能 递归实现n\*(n+1)的求解，$a0作为参数输入，程序结束恢复$a0的值