1、 SUB R3，R2，R1，LSR R0写出上面指令的32位编码，并说明每一位（或几位）的含义。格式可以按下表来写

该指令为R3=R2-R1>>R0，32编码为1110 0000 0100 xxxx xxxx xxxx x010 xxxx，每一位的含义如下表所示

|  |  |  |
| --- | --- | --- |
| 指令编码（从高位到低位） | 位数 | 含义 |
| 1110 | 31-28 | 条件位，为默认值AL，表示任何时候都执行本指令 |
| 000 | 27-25 | 表示第一个操作数是寄存器类型 |
| 0010 | 24-21 | 减操作码SUB |
| 0 | 20 | 可选后缀S，无表示不更新CPSR中的条件码标志位 |
| R2 | 19-16 | 操作数1寄存器编码（R2） |
| R3 | 15-12 | R0 |
| R0 | 11-7 | 操作数2（R1）的移位长度，为R0 |
| 01 | 6-5 | 移位类型，逻辑右移操作码（LSR） |
| 0 | 4 | 保留位 |
| R1 | 3-0 | 操作数2寄存器编码（R1） |

2、读程序，给每一行增加注释，并说明执行过程中R0,R1,R2的值如何变化

AREA StrCopy, CODE, READONLY ; 定义代码段 StrCopy，只读  
 ENTRY ; mark the first instruction to call  
  
start  
X EQU 88 ; 定义符号常量 X = 88  
Y EQU 76 ; 定义符号常量 Y = 76  
Z EQU 96 ; 定义符号常量 Z = 96  
STACK\_TOP EQU 0X1000 ; 定义符号常量 STACK\_TOP = 0x1000  
  
 MOV R0,#0XAB ; 将 0xAB 装载到 R0 寄存器  
LOOP1 MOV R0,R0,ASR#1 ; R0右移一位，将结果存回 R0  
 CMP R0,#0X50 ; 比较 R0 和 0x50  
 BGE LOOP1 ; 如果 R0 大于或等于 0x50，则跳转到 LOOP1  
 MOV R1,#Y ; 将 Y 装载到 R1 寄存器  
 ADD R2,R0,R1,LSL #1 ; 将 R0 和 左移一位的R1 相加，结果存入 R2  
 MOV SP,#0X1000 ; 将 0x1000 装载到 SP 寄存器  
 STR R2,[SP] ; 将 R2 寄存器的值存入 SP 指向的内存地址！！  
 MOV R0,#Z ; 将 Z 装载到 R0 寄存器  
 AND R0,R0,#0XFF ; 将 R0 寄存器的值与 0xFF 按位与，结果存入 R0  
 MOV R1,#Y ; 将 Y 装载到 R1 寄存器  
 ADD R2,R0,R1,LSR #1 ; 将 R0 和 R1 右移一位相加，结果存入 R2  
 LDR R0,[SP] ; 将 SP 指示的内存地址中的值加载到 R0 寄存器  
 MOV R1,#0X01 ; 将 0x01 装载到 R1 寄存器  
 ORR R0,R0,R1 ; 将 R0 寄存器的值与 R1 寄存器的值按位或，结果存入 R0  
 MOV R1,R2 ; 将 R2 寄存器的值移入 R1 寄存器  
 ADD R2,R0,R1,LSR #1 ; 将 R0 和 R1 右移一位相加，结果存入 R2  
STOP B STOP ; 无条件跳转到 STOP 标签处  
 END

执行过程中，R0、R1、R2 的值变化如下：

* MOV R0, #0XAB，R0 = 0xAB
* MOV R0, R0, ASR #1，右移 R0 的值一位，结果存回 R0
* CMP R0, #0X50，比较 R0 和 0x50
* MOV R1, #Y，R1 = 76
* ADD R2, R0, R1, LSL #1，R2 = R0 + (R1 左移一位)
* MOV SP, #0X1000，SP = 0x1000
* STR R2, [SP]，将 R2 的值存入 SP 指示的内存地址
* MOV R0, #Z，R0 = 96
* AND R0, R0, #0XFF，R0 = R0 与 0xFF 的按位与结果
* MOV R1, #Y，R1 = 76
* ADD R2, R0, R1, LSR #1，R2 = R0 + (R1 右移一位)
* LDR R0, [SP]，将 SP 指示的内存地址中的值加载到 R0
* MOV R1, #0X01，R1 = 1
* ORR R0, R0, R1，R0 = R0 与 R1 的按位或结果
* MOV R1, R2，R1 = R2
* ADD R2, R0, R1, LSR #1，R2 = R0 + (R1 右移一位)