1.(P.112 4)请依据 a 中的反汇编代码，将 b 中对应的 C 语言代码补充完全。

```
p112.4
1   # void cond(int a,int *p)
2   #a in %edi,p in %esi
3       cond:
4           testl %esi,%esi  #检测esi,如果p==0,设置标志位ZF=1
5           je .L1           #如果等于ZF=1,则跳转到L1
6           cmpl %edi,(%esi) #比较p指向内存的值和edi中的值的大小
7           jge .L1          #如果p指向内存处的值大于等于a则跳转到L1
8           movl %edi,(%esi) #a的值写入p指向内存处
9       .L1:
10          ret              #弹出返回地址
```

解释说明：

（1）testl 指令

- **功能**：对两个操作数执行按位与（AND）操作，但不保存结果，仅影响标志寄存器。
- **标志位影响**：
  - SF（**符号标志**）：结果的最高位（符号位）。
  - ZF（**零标志**）：若结果为 0，则置 1；否则置 0。
  - PF（**奇偶标志**）：结果低 8 位的奇偶性（1 的个数是否为偶数）。
  - CF（**进位标志**）和 OF（**溢出标志**）：均被清零（置 0）。

（2）je 指令

- **功能**：当 ZF=1 时跳转（即两值相等或运算结果为 0 时跳转）。

（3）cmpl 指令

- **功能**：比较两个操作数，等价于 destination - source，但仅设置标志位。

- **标志位影响：**
  - **CF（进位标志）：**无符号数比较中，若 destination < source 则置 1。
  - **OF（溢出标志）：**有符号数比较中，若结果溢出则置 1。
  - **SF（符号标志）：**结果的最高位（有符号数的符号）。
  - **ZF（零标志）：**若差值为 0 则置 1。

（4）**jge 指令**

- **功能：**当 **SF=O 且 ZF=1** 时跳转（即有符号数比较中 destination ≥ source）。

（5）**ret 指令**

- **功能：**从子程序返回，弹出栈顶地址作为返回地址。

由汇编补充的 c 代码：

```c
//由此对应的C语言代码
void cond(int a,int *p){
    if(p&&*p<a){ //p非空，p指向的值小于a，将a覆盖p指向的内容
        *p=a;
    }
}
```

2. （P112 5）有一段汇编代码如下，试画出其逻辑流程图并写出对应该流程图的 C 语言 goto 版本，变量名可自行拟定。

work > work3 > ASM p112_5.s

```
1   movl 0x0,%eax   #eax=0
2   cmpl %edx,0     #比较0和edx中值的大小
3   jz    .L2       #如果edx=0，则跳转到L2
4   jg    .L1       #如果0>edx，则跳转到L1
5   movl 0xffffffff,%eax #eax=0xffffffff
6   jmp   .L2       #无条件跳转到L2
7   .L1:
8       movl 0x1,%eax #eax=1
9   .L2:
10      ret #弹出返回地址
```

解释说明：

（1）jz

- **跳转条件**：当**零标志位（ZF）**被置位（ZF=1）时跳转。

（2）jg

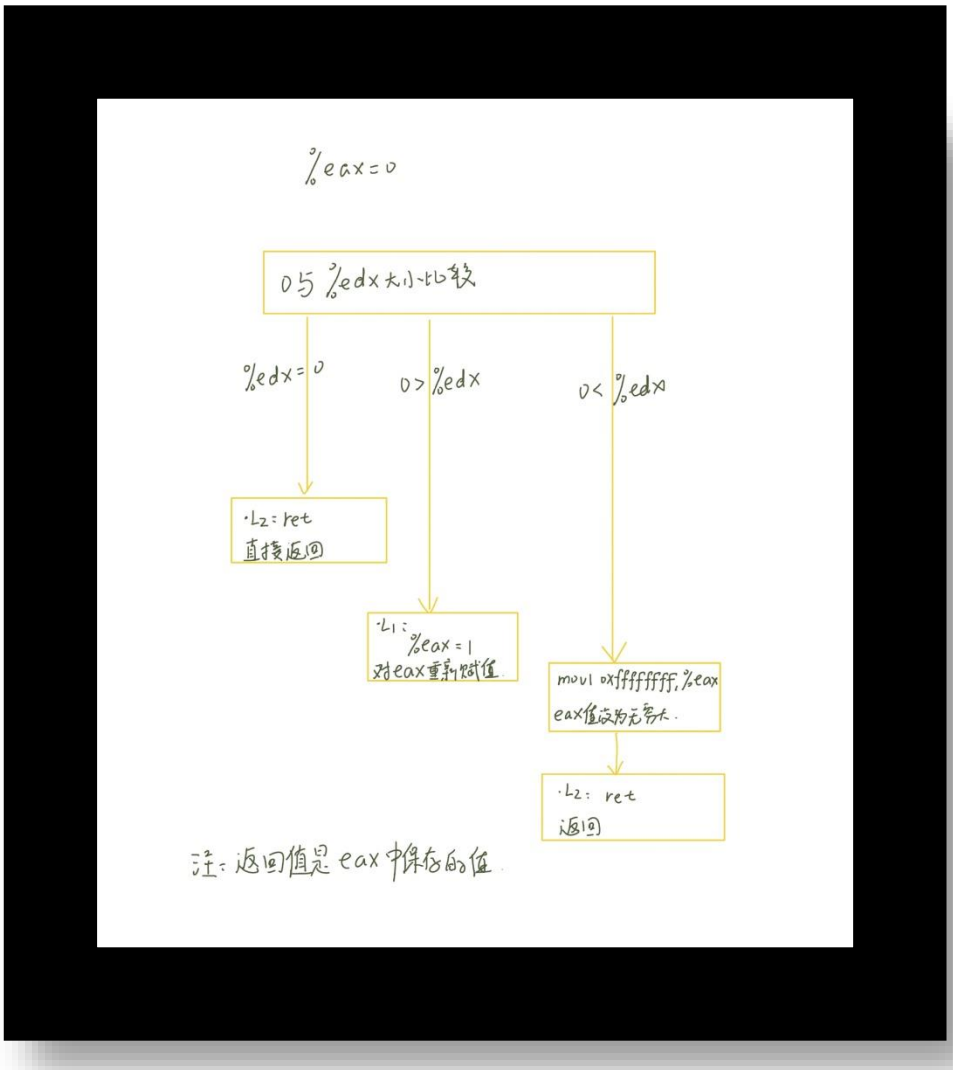
- **跳转条件**：当**有符号数比较结果为大于**时跳转。

  具体条件：SF=OF 且 ZF=0（即符号位与溢出位相同，且结果非零）

（3）jmp

- **跳转条件**：无条件跳转，无论标志寄存器状态如何。
- **依赖的标志位**：不依赖任何标志位。

逻辑流程图如下：



对应该流程图的 C 语言 goto 版本：

work > work3 > C p112_5.c > ...

```c
const MAX=0xffffffff;
int func(int a,int d){
    a=0;
    if(0==d){
        goto L2;
    }
    else if (0>d)
    {
        goto L1;
    }else{
        a=MAX;
        goto L2;
    }
L1:
    a=1;
L2:
    return a;
}
```

3.（P112 6）已知有下面一段不完整的 C 语言代码及其对应的汇编代码，请依据汇编代码将 C 语言代码填写完整。

work > work3 > C p112_6.c > puzzle(unsigned)

```c
unsigned puzzle(unsigned n){
    if(){
        return 1;
    }else{
        return 1+puzzle();
    }
}
```

work > work3 > p112_6.s

```
1   pushl %ebp              #保存旧的基址指针（caller 的 ebp）
2   movl %esp, %ebp         #设置新的基址指针（当前栈顶为函数栈帧起点）
3   subl $8, %esp           #为局部变量分配8字节空间（栈向下增长）
4
5   cmpl $0, 8(%ebp)        #比较参数（8(%ebp)）与0
6   jne .L2                 #如果参数 != 0，跳转到.L2
7   movl $1, -4(%ebp)       #如果参数 == 0，局部变量-4(%ebp) = 1
8   jmp .L3                 #无条件跳转到函数结尾
9
10  .L2:
11      movl 8(%ebp), %eax    #将参数（8(%ebp)）加载到 EAX
12      shrl $3, %eax         #eax右移3位（等价于无符号除法：参数/8）
13      movl %eax, (%esp)     #将eax的值作为参数压栈（调用puzzle函数）
14      call puzzle           #调用puzzle函数，返回值存储在eax
15      addl $1, %eax         #eax= puzzle(参数/8)+1
16      movl %eax, -4(%ebp)   #局部变量-4(%ebp)=计算结果
17
18  .L3:
19      movl -4(%ebp), %eax   #将局部变量-4(%ebp)的值加载到eax
20      leave                 #恢复栈帧：等价于movl %ebp,%esp + popl %ebp
21      ret                   #返回调用者
```

解释说明：

（1）subl

功能：执行有符号减法：dest = dest - src，结果存储在 dest 中。

（2）jne

功能：条件跳转指令：当**零标志位（ZF 为 0**）时跳转到目标地址。

（3）shrl

功能:逻辑右移：将寄存器或内存中的值向右移动 n 位，左侧补 0。

（4）addl

功能：执行有符号加法：dest = dest + src，结果存储在 dest 中。

（5）leave

功能：恢复栈帧，通常用于函数退出前清理栈。等价于：

```
movl %ebp, %esp    ; 将栈指针 esp 恢复到基址指针 ebp 的位置
popl %ebp          ; 弹出旧的基址指针到 ebp
```

（6） call

功能:调用子程序，将返回地址压栈并跳转到目标地址。

上述 C 代码补充完整后：