说明：1）允许使用伪指令；2）通过调用名称引用寄存器；3）按顺序直接写出答案即可。

寄存器	调用名字
x0	zero
x1	ra
x2	sp
x3	gp
x4	tp
x5-x7	t0-t2
x8	s0/fp
x9	s1
x10-x11	a0-a1
x12-x17	a2-a7
x18-x27	s2-s11
x28-x31	t3-t6

一、计算(35+46-17)*32，结果存入寄存器 s0。

参考程序如下，请在其中的空白处填上合适的指令：

```
addi  s0,zero,35          ；s0 初始化为 35
【addi  s0,s0,46】         ；s0 加 46
addi  s0,s0,-17           ；s0 减 17
【addi  s1,zero,32】       ；s1 初始化为 32
mul  s0,s0,s1             ；s0 乘 32
```

二、判断寄存器 t0 中的有符号数（直接初始化）的符号：如果为正，将寄存器 s0 置 1；如果为负，将 s0 置-1；否则，将 s0 清 0。

参考程序如下，请在其中的空白处填上合适的指令：

```
        addi  t0,zero,23          ；初始化 t0
        【beq  t0,zero,label1】    ；t0 为 0 时转移到 label1 处
        bltz  t0,  label2         ；t0 小于 0 时转移至 label2 处
        【addi  s0,zero,1】        ；t0 大于 0 时，s0 置 1
        jal  zero,ok              ；转移至所有分支结束处（汇合点）
label1: 【addi  s0,zero,0】        ；t0 等于 0 时，s0 置 0
        jal  zero,ok              ；转移至所有分支结束处（汇合点）
```

```
label2: 【addi s0,zero,-1】          ；t0 小于 0 时，s0 置-1
    ok: ......                      ；所有分支结束处（汇合点）
```

三、采用循环结构按 16 进制分离出 10 进制数 1234 各位上的数，并将分离结果保存到堆栈（低位先入栈）。

参考程序如下，请在其中的空白处填上合适的指令：

```
        addi  s0,zero,1234          ；s0 初始化为 1234
    on: 【beq  s0,zero,done】       ；s0 为 0 时，分离过程结束
        andi  s1,s0,15              ；截取 s0 的低 4 位（高位清 0），置于 s1
        【addi  sp,sp,-1】          ；申请一个字节的堆栈空间
        sb  s1,0(sp)                ；s1 的低位字节入栈
        【srli  s0,s0,4】           ；s0 逻辑右移 4 位
        jal  zero,on                ；构造循环
  done: ……
```

四、以 Intx 为首地址的字类型整数组中依次存放着 10 个有符号整数，将其中最小的整数存入寄存器 s0。

参考程序如下，请在其中的空白处填上合适的指令：

```
        la  t0,Intx                 ；t0 初始化为数组首地址
        【lw  s0,0(t0)】            ；读第 1 个整数，置于 s0
        addi  t0,t0,4               ；修改 t0，指向第 2 个整数
        【addi  t1,zero,9】         ；初始化循环计数器 t1
  loop: lw  s1,0(t0)                ；读下一个整数，置于 s1
        【blte  s0,s1,label1】      ；s0 小于等于 s1 时，不用更新 s0
```

```
        add  s0,zero,s1          ；用 s1 更新 s0
label1: 【addi  t0,t0,4】        ；修改指针 t0
        addi  t1,t1,-1           ；修改计数器
        【bne  t1,zero,loop】    ；构造循环
```

五、设 xyz 为字类型的整数组，共 10 个元素，用子程序 sum 求这个数组的和并将求得的和存入字变量 result。

参考程序如下，请在其中的空白处填上合适的指令：

```
main: la  a0,xyz                 ；准备入口参数 a0：首地址
      【addi  a1,zero,10】       ；准备入口参数 a1：长度
      jal  ra,sum                ；调用
      【la  t0,result】          ；准备 result 的地址
      sw  a2,0(t0)               ；写出口参数 a2
      ……
 sum: add  a2,zero,zero          ；初始化 a2
  on: 【lw  t0,0(a0)】           ；读数据
      add  a2,a2,t0              ；累加
      【addi  a0,a0,4】          ；修改地址
      addi  a1,a1,-1             ；修改计数器
      【bne  a1,zero,on】        ；构造循环
      jalr  zero,0(ra)           ；返回
```

六、设 xyz 为字类型整数组，以 0 作为结束标记。请采用递归子程序 Inverse 按逆序输出（直接调用子程序 Print，入口参数：a0=要输出的整数）这个整数组。

参考程序如下，请在其中的空白处填上合适的指令：

```
main: la a0,xyz                    ；准备入口参数
      【jal ra,Inverse】            ；调用
      ……
Inverse: lw t0, 0(a0)              ；读第 1 个整数
      【beq t0,zero,return】        ；第 1 个整数为 0 时，直接返回
      addi a0,a0,4                 ；修改地址，指向第 2 个整数
      【addi sp,sp,-8】             ；申请堆栈空间
      sd ra,0(sp)                  ；备份返回地址
      【addi sp,sp,-8】             ；再申请堆栈空间
      sd t0,0(sp)                  ；备份第 1 个整数
      【jal ra,Inverse】            ；递归
      ld a0,0(sp)      ；恢复第 1 个整数，同时为调用 print 准备入口参数
      【addi sp,sp,8】              ；释放堆栈空间
      jal ra,print                 ；调用 print
      【ld ra,0(sp)】               ；恢复返回地址
      addi sp,sp,8                 ；释放堆栈空间
return: 【jalr zero,0(ra)】         ；返回
```

说明：1）根据算法思路和前后逻辑确定每一条指令，尤其是要用到的寄存器。

2）**连续**多次申请堆栈空间、备份数据时，也可以一次性申请空间并依次备份；恢复数据并释放空间也是如此。

3）恢复数据时，未必一定要恢复到原来的数据空间！

4）有规划的生活更美好！有规划的编程更容易！