RISC-V汇编练习2

总分: 60

*此封面页请勿删除，删除后将无法上传至试卷库，添加菜单栏任意题型即可制作试卷。本提示将在上传时自动隐藏。

多选题 2分 设置

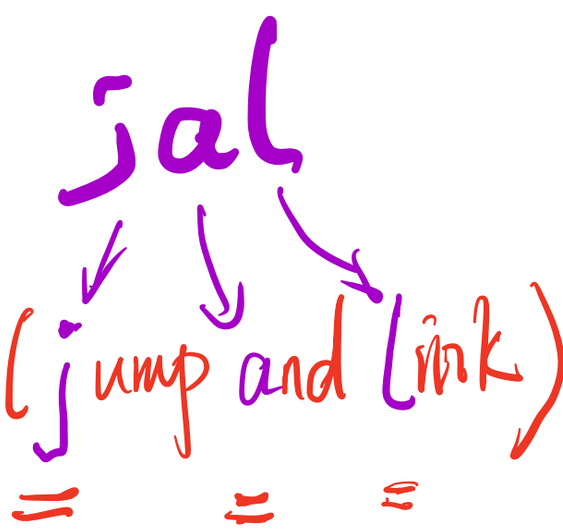
1、RV伪指令 **j Label** 以及伪指令**jr x1**的原型分别是_____

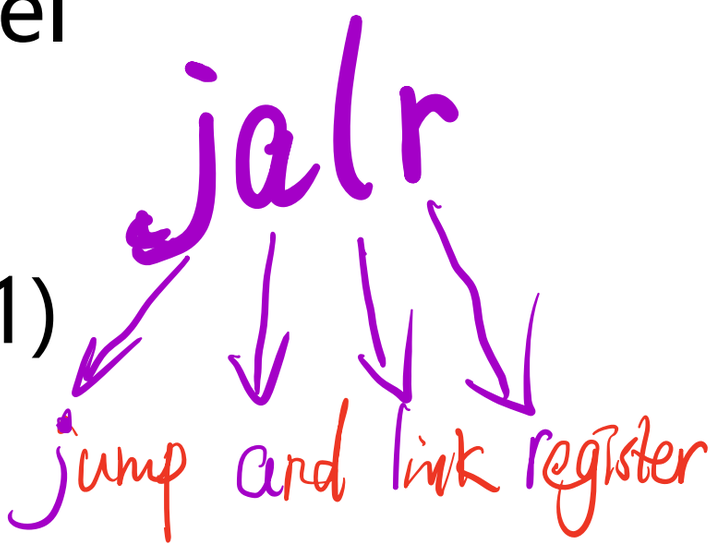
A jal x0, Label

B jal x1, Label

C jalr x0, 0(x1)

D jalr x1, 0(x1)

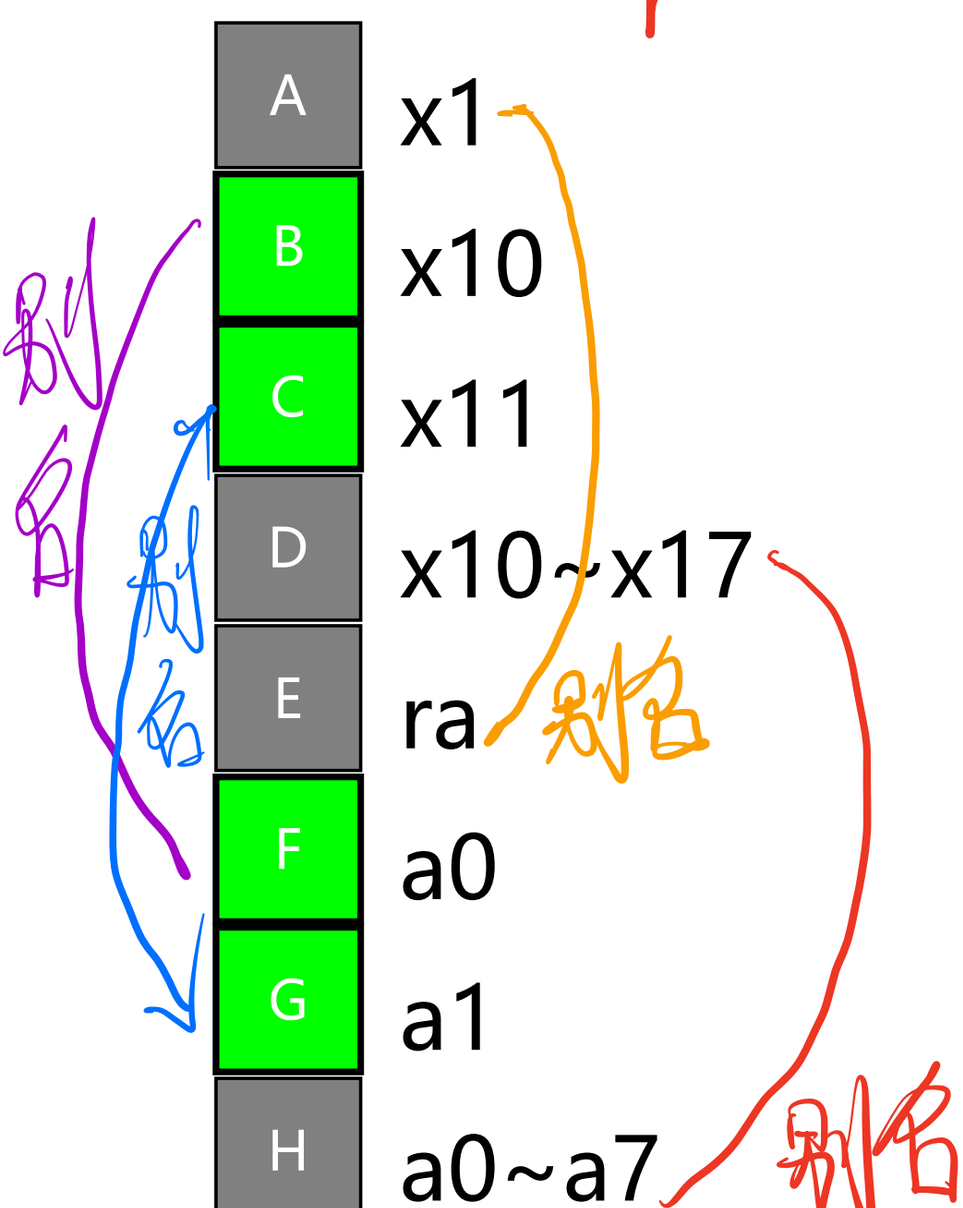




多选题 4分 设置

2、以下寄存器中，哪些寄存器既用来存函数参数，又用来存函数返回值 x0和x1。

即：a0和a1



多选题 2分

设置

3、关于指令beq x0, x0, Label 与指令jal x0, Label 说法正确的是：

A 它们可以等价替换

仅当A可以的跳小时才可以替换

B PC间接寻址范围不同

C 指令的立即数表示范围不同

规整的指令编码

- 所有通用寄存器在指令码的位置是一样的，方便译码；
- 所有的指令都是32位字长，有 6 种指令格式：寄存器型，立即数型，存储型，分支指令、跳转指令和大立即数

R 型	funct7	rs2	rs1	funct3	rd	opcode
I 型	imm[11:0]		rs1	funct3	rd	opcode
S 型	Imm[11:5]	rs2	rs1	funct3	imm[4:0]	opcode
SB / B 型	Imm[12,10:5]	rs2	rs1	funct3	imm[4:1,11]	opcode
UJ / J 型	Imm[20,10:1,11,19:12]				rd	opcode
U 型	Imm[31:12]				rd	opcode

填空题 13分

设置

1）RV函数调用中，ra表示**函数返回地址（Return Adress）**，其对应的寄存器是 [填空1] 。

x1

2）**a0和a1**既可以**存储函数参数，又可以存储函数返回值**，它们对应的寄存器是分别是 [填空2] 和 [填空3] 。

x10 x11

3）查**卡片1**，下面指令分别是：

sub属于 [填空4] 型指令
addi属于 [填空5] 型指令
ld 属于 [填空6] 型指令
sd 属于 [填空7] 型指令
jal 属于 [填空8] 型指令
jalr 属于 [填空9] 型指令
beq属于 [填空10] 型指令

R I S J B

（本题填空限制在：R、I、S、J、B）

填空题 4分 设置

4）执行下列指令序列后，寄存器x10的值是 [填空1] ？（填写十进制的值）。

```
addi x10, x0, 4
bge x10, x0, L1
    srli x10, x10, 1
    j Exit
L1:
    slli x10, x10, 2
Exit:
```

① x10=4
② x10>=x0跳转L1
③ x10=x10×4=16
④ 结束

执行下列指令序列后，寄存器x10的值是多少？

```
addi x10, x0, 4
bge x10, x0, L1
    srli x10, x10, 1
    j Exit
L1:
    slli x10, x10, 2
Exit:
```

解析：
1：执行后x10=4
2：判断x10的值是否大于等于0？如果大于等于跳转L1，否则，顺序执行。
这里x10=4大于0，跳转L1执行。
L1：x10寄存器逻辑左移2位，执行后x10=16
Exit：退出程序
综上，执行下列指令后，x10的值为16

填空题 5分 设置

5） 执行下列指令序列后，寄存器x10的值是 [填空1] ？（填写十进制的值）。

```
addi x11, x0, 11
mv x5, x0
mv x10, x0
Loop:
    bge x5, x11, Done
    add x10, x10, x5
    addi x5, x5, 1
    j Loop
Done:
```

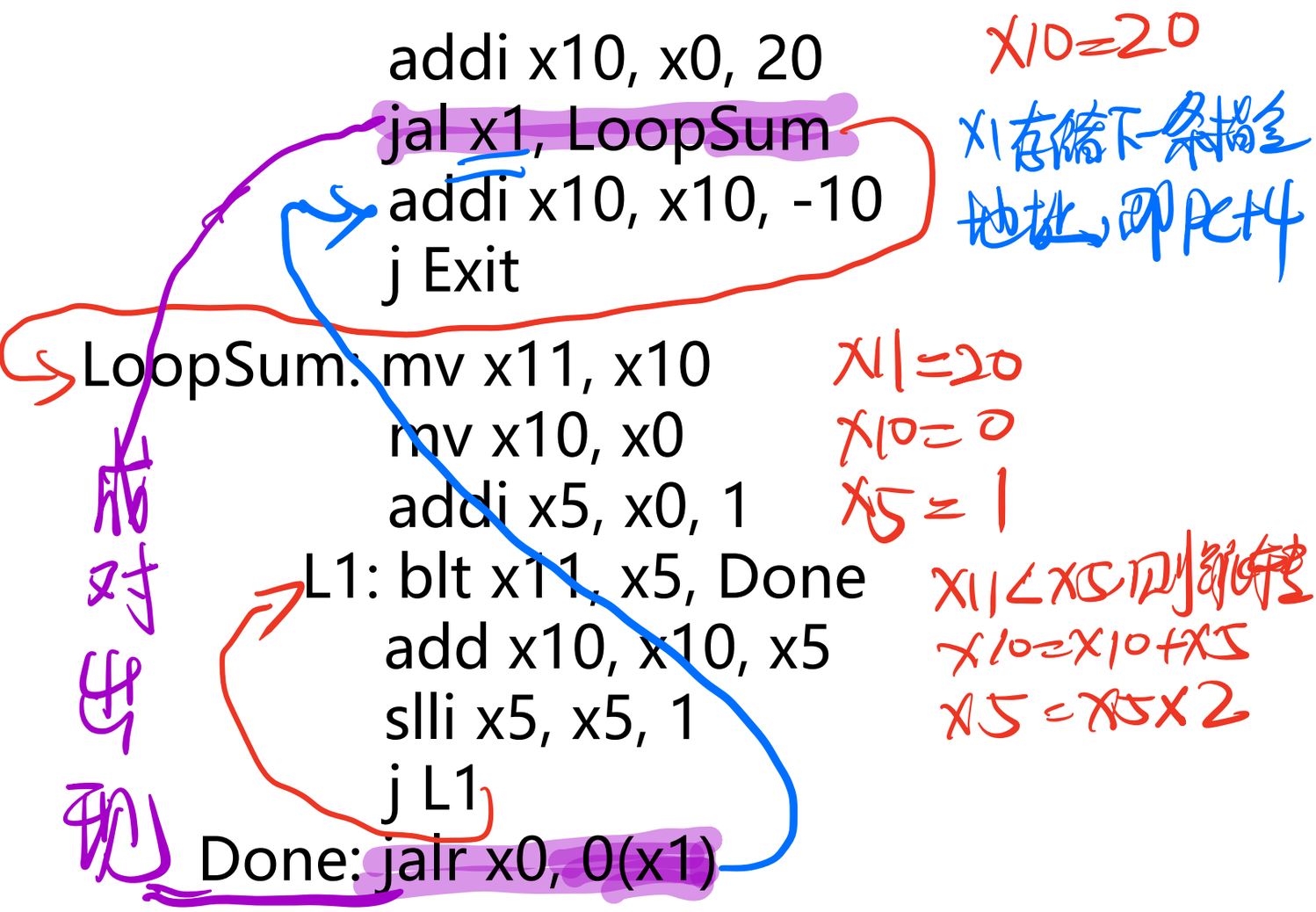
执行下列指令序列后，寄存器x10的值是多少？

```
addi x11, x0, 11
mv x5, x0
mv x10, x0
Loop:
    bge x5, x11,
    Done
    add x10, x10, x5
    addi x5, x5, 1
    j Loop
Done:
```

解析：
整个程序的流程相当于使用for循环求1+2+3+...+10的值，结果存入x10寄存器。
1：执行后x11=11
2、3：使用mv指令将x5和x10寄存器清0。
Loop:
判断寄存器x5的值是否大于等于x11的值(11)如果大于则跳转Done，程序执行完成。否则，累加寄存器x10的值，x10 = x10 + x5。然后将寄存器x5的值+1，无条件跳转到循环起始位置，开始下一轮循环。
最终x10 = 1+2+3+...+10 = 55

6）执行下列指令序列后，寄存器x10的值是 [填空1]？（填写十进制的值）。

```
        addi x10, x0, 20
        jal x1, LoopSum
        addi x10, x10, -10
        j Exit
LoopSum: mv x11, x10
        mv x10, x0
        addi x5, x0, 1
    L1: blt x11, x5, Done
        add x10, x10, x5
        slli x5, x5, 1
        j L1
 Done: jalr x0, 0(x1)
  Exit:
```



x10=1+2+4+8+16=31

当x5=32时，跳转到Done

函数返回到addi x10,x10,-10

此时 x10=x10-10=31-10=21

jal x1, Label 与 jalr x0, 0(x1) 成对出现，从而实现函数调用

执行下列指令序列后，寄存器x10的值是多少？

```
addi x10, x0, 20
jal x1, LoopSum
addi x10, x10, -10
j Exit
LoopSum:
        mv x11, x10
        mv x10, x0
        addi x5, x0, 1
        L1:
                blt x11, x5, Done
                add x10, x10, x5
                slli x5, x5, 1
                j L1
        Done:
                jalr x0, 0(x1)
Exit:
```

解析：
首先我们先梳理一下整个程序，LoopSum函数的功能是求小于x10寄存器的所有2的指数项的值。
1-2：我们给x10赋值20，然后调用LoopSum函数，调转到第6行执行，返回地址为addi x10,x10,-10指令的地址。
6-9：初始化函数所需的寄存器，由于函数的计算结果需要由x10寄存器的值返回，所以我们使用mv指令将x10寄存器的值赋给x11，并且将x10的值清0，同时我们将x5的值初始化为1。
L1循环体：首先判断x11的值(20)是否小于当前x5寄存器的值，如果小于则跳转Done，否则，累加x10=x10+x5。将寄存器x5的值左移1位，相当于x5=x5*2，最后跳转到循环起始位置开始下一轮循环。
跳转Done后：使用一条无条件跳转指令从函数中返回。
从函数返回后：将得到的计算结果x10再减去10，得到最终的结果，然后退出程序。

解析：
基于上述的梳理，我们可以得到，因为我们首先将x10赋值20，所以LoopSum函数计算20以内所有2的指数项的和。故从函数LoopSum中返回后，寄存器x10的值为：
x10 = 1+2+4+8+16 = 31
之后我们又将x10的值减去了10，所以x10寄存器的最终值为：
x10= 31-10 = 21

填空题 0分 设置

此题未设置答案，请点击右侧设置按钮

1）RV汇编语言的传参的基本原理？

答：利用寄存器x10~x17(即a0~a7)在主调函数和被调函数之间进行存取。

寄存器相当于中转站，主调函数和被调函数共享这些寄存器。

此题未设置答案，请点击右侧设置按钮

2）RV汇编语言的函数返回值如何传递?

答：被调函数将返回值存在寄存器x10或x11中，主函数即可获取。因为这些寄存器是主调函数和被调函数共享的。

注意：x10和x11具有传参数和返回值的双重角色。

填空题 0分 设置

此题未设置答案，请点击右侧设置按钮

3）被调函数完成后返回主调函数的原理是？

答：主调函数利用jal x1, Label进行函数跳转同时将主调函数的下一条指令地址存储在x1里；被调函数执行完，用jalr x0, 0(x1）返回主调函数继续执行jal后面的那条指令，即执行x1里存的地址对应的指令。

填空题 0分 设置

此题未设置答案，请点击右侧设置按钮

4）jal和jalr的区别是？

答：jal 跳转的地址是PC相对寻址，原始形式是PC+offset；语法是跳转到Label对应的指令地址。

jalr跳转方式是基址寻址：即寄存器里存的数作为基地址再加上偏移量，跳转范围由寄存器位数决定。

两个指令格式中的第一个参数功能是相同的，默认存储PC+4。**实际函数调用中，jal x1, Label与jalr x0, 0(x1)往往是配对出现的。**

5）为什么要用栈存储和恢复寄存器内容?

答：解决寄存器使用冲突。主从函数在调用过程中可能使用相同的寄存器，导致主调函数的寄存器内容被覆盖，比如递归函数中的x1和x10，调用中会被不断覆盖，因此每一层都需要不断保存用到的寄存器。也遵循caller和callee规则由主调函数和被调函数分别保存。