实验题目

题目1 - 小试牛刀

请计算c=a^b，其中a为整数，b为大于等于0的整数。

a和b的初始值存储在从x1000 0000开始的一段连续的存储单元中，c存储在b之后。

要求：

1. 将a^b的结果保存在c的地址中；
2. 使用系统调用输出c的结果到控制台: "c = xxx"，其中xxx为你的计算结果；
3. 你的程序应该正常执行并返回0。

提示：

1. 你的.data段应该为：

```
.data
    a: .word 2
    b: .word 3
    c: .space 4
    str: .string "c = “
```

2. 关于系统调用参考实验环境手册。

   输出“c = xxx"，可以先输出"c = " (PrintString)，再输出"xxx" (PrintInt)

题目2 - 数组处理

将数组中的正数-3，负数*2，0替换为114514并存回原存储单元之中。

假设这一整数数组存储于从x1000 0000开始的一段连续的存储单元之中。

要求：

1. 要求存回原存储单元之中；
2. 你的程序应该正常执行并返回0。

提示：

你的.data段应该为：

```
.data
    array: .word 1, -12, 0, -67, 99, -23, 2024, 58, 21
```

题目3 - 子串验证

给定字符串string和子串substring，判断substring是否是string的子串。

string和substring存储在从x1000 0000开始的一段连续的存储单元中。

子串：连续的子序列，举例来说"Cs"是"ILoveCs"的子串，但"Is"并不是"ILoveCs"的子串。

要求：

1. 若substring是string的子串，向控制台输出“Substring:)”，反之，输出"NotSubstring:("
2. 你的程序应该正常执行并返回0。

提示：

1. 你的.data段应该为：

   ```
   .data
       string: .asciz "ILoveCs"
       substring: .asciz "Cs"
       findstr: .string "Substring:)"
       notfindstr: .string "NotSubstring:("
   ```

   可以自己多测试一些样例，如：“ILoveCs" - "Css"

2. 注意按序读取字符串时，地址增长的大小以及load的指令的选择。

题目4 - 函数调用

请将下面的C语言程序转化为RISC-V汇编代码。

```
int x = 1;
int y = 2;
int z;

int triple(int a) {
    return a + a + a;
}

int minus(int a, int b) {
    return a - triple(b);
}

int main() {
    z = minus(x, y);
    return 0;
}
```

要求：

请在给定的汇编代码框架下补充完整，注意请运用函数调用。(如有需要，可以添加标识符)

```
.data
    x: .word 1
    y: .word 2
    z: .space 4

.text
```

```
.globl main
triple:
    # ...
    ret
minus:
    # ...
    ret
main:
    # ...
    jal minus
    # ...
```

注意你的程序应该正常执行并返回0；记得将结果写入z所在地址。

提示：注意使用栈保存你的返回地址和参数。

题目5 - 递归调用

请将以下C语言函数转化为RISC-V汇编代码。

```c
unsigned int fibonacci(unsigned int n) {
    if (n == 0) return 1;
    if (n == 1) return 1;
    return f(n-1) + f(n-2);
}
```

要求：

请在给定的汇编代码框架下补充完整，注意递归调用。(如有需要，可以添加标识符)

```
.globl main

fibonacci:
    # ...
    # n >= 2 => jump to _recurse
    # ...


    # return 1;
    # ...
    ret

# this is a branch
_recurse:
    # calculate fibonacci(n-1)
    # ...

    # calculate fibonacci(n-2)
    # ...

    # fibonacci(n-1) + fibonacci(n-2)
    # ...
    ret
```

```
main:
    li  a0, 15          # Input will be 15
    jal fact           # a0 = fibonacci(15);

    li a7, 1           # print a0
    ecall

    li a7, 10          # return 0
    ecall
```

提示：注意使用栈保存你的返回地址和参数。