# 基于RISC-V的FreeRTOS国密算法功能实现

## 1. 任务目标

- 基于RISC-V架构在freertos实现国密算法，实现对数据的加密解密
- 实现基于freertos的SM3、SM4软算法
- 编写测试用例，覆盖所有代码执行路径
- 性能测试与优化

## 2. 任务分析

### 2.1 RISC-V指令集

RISC-V是一套开源免费指令集(Instruction set architecture,ISA),

RISC-V属于**精简指令集**(RISC)的一种，RISC指令特定是所有指令的长度固定(32位，可扩展到任意16倍长)，不同指令的执行时间相等，因此可以通过比较编译后的指令数量计算程序执行时间。

与X86不同的是RISC-V架构采用**Load-Store 结构**，RISC-V架构中CPU不会对内存中的数据进行操作，所有的计算都要求在寄存器中完成，而寄存器和内存的通信则由单独的指令来完成。例如，RISC-V指令里的ADD指令要求两个操作数都在寄存器中，但X86指令操作数可以在内存中。既RISC-V不支持内存寻址。

RISC-V规范里定义了有32位与64位两种，但嵌入式系统中一般是使用**32位**，RISC-V**支持SIMD**指令

### 2.2 FreeRTOS实时操作系统

FreeRTOS是一个多**任务实时操作系统**, 任务类似于分时操作系统中进程的概念，FreeRTOS调度器根据优先级决定当前应该运行的任务，任意时刻只有一个任务运行，调度器通过任务句柄管理任务，类似于进程的PCB.

### 2.3 国密算法

SM3 是Hash算法的一种其输出长度是256位，Hash算法是一种将任意长度的输入映射到固定长度输出的不可逆单向算法

SM4 是一种分组对称加密算法，分块长度是128位

算法具体流程见附件

## 2.4 实现原理

原理就算**将C/C++代码编译移植到对应指令集的对应平台上**(RISC-V, freeRTOS)

众所周知，C程序转为被操作系统识别的可执行文件需要经过**编译和链接**两个阶段。

编译阶段其目标是 : 将C 源码翻译成对应指令集上的二进制指令集，编译阶段只跟目标程序使用指令集有关

链接阶段其目标是 : 组装编译阶段生成模块为一个完整可执行程序，并用特定格式排版二进制指令，如WINDOWS使用PE,UNIX程序使用ELF，排版的目的是为了被操作系统所识别以至于能将这些指令作为一个进程(分时操作系统)加载到内存中，CPU最终从内存取指令译码执行。

例如以下C程序:

```
#include <stdio.h>

int main(){
    puts("Hello World\n");
    return 0;
}
```

使用X86-64 GCC编译器编译后生成的汇编指令:

```
.LC0:
        .string "Hello World\n"
main:
        push    rbp
        mov     rbp, rsp
        mov     edi, OFFSET FLAT:.LC0
        call    puts
        mov     eax, 0
        pop     rbp
        ret
```

使用RISC-V 32位指令编译后的汇编指令为:

```
.LC0:
        .string "Hello World\n"
main:
        addi    sp,sp,-16
        sw      ra,12(sp)
        sw      s0,8(sp)
        addi    s0,sp,16
        lui     a5,%hi(.LC0)
        addi    a0,a5,%lo(.LC0)
        call    puts
        li      a5,0
        mv      a0,a5
        lw      ra,12(sp)
        lw      s0,8(sp)
        addi    sp,sp,16
        jr      ra
```

但是编译后汇编指令并不能直接运行，因为puts函数在该模块中并无定义，需要链接到其他模块才能形成一个完整程序。

puts是一个C标准库的函数，C编译器为程序默认链接到C标准库因此使得程序能直接运行。

但不同操作系统对C标准库的实现不一样，这是因为C标准库里会去调用系统接口，不同操作系统对系统接口实现不同使得C标准库的实现也不相同， Windows实现的C标准库叫CRT, LINUX系统实现的C标准库叫GLIBC,Android则使用musl库， 因此，就算在相同指令集下不同操作系统的二进制程序也不能混用。

有一种情况是程序的实现可以不依赖于C标准库，这被称为**独立实现**(freestanding)，独立实现的好处是在不修改源码的情况下只需要重新编译就能将程序移植到任意平台，C++独立实现可使用的C标准库功能可以参考https://en.cppreference.com/w/cpp/freestanding

# 3. 任务流程

## 3.1 基于RISC-V的freeRTOS开发环境搭建

- 安装RISC-V gcc工具链(见附件)
- **下载freeRTOS源码后使用gcc工具链编译安装到开发板**
- 可以搭建freeRTOS虚拟机(QEMU)用于快速测试程序正确性(https://www.freertos.org/RTOS-RISC-V-FreedomStudio-QMEU.html)
- 基于freeRTOS开发者文档编写hello world程序编译运行

## 3.2 国密算法编写

- **使用C/C++ freestanding模式实现SM3算法，SM4 算法**
- 使用freeRTOS提供的C库实现SM3与SM4的多包运算
- 编写**测试用例**测试算法正确性与程序健壮性

### 3.3 算法性能测试

- 使用Tracealyzer分析算法性能
- 根据结果调整源码优化性能，适当的情况下使用内联汇编优化算法性能

## 4. 参考链接


1. https://freertos.org/
2. https://en.wikipedia.org/wiki/RISC-V
3. https://zhuanlan.zhihu.com/p/96531591?ivk_sa=1024320u
4. https://blog.csdn.net/wangyijieonline/article/details/109677855
5. https://github.com/kendryte/kendryte-freertos-demo