

山东大学



计算机系统原理实验报告

组员 1：谢亦凡、组员 2：韩甜甜、组员 3：董晓冉、组员 4：朱科睿

组员 1 学号：	202300460106	班级：	网安二班
组员 2 学号：	202300460141	班级：	网安二班
组员 3 学号：	202200210034	班级：	密码一班
组员 4 学号：	202300460035	班级：	网安一班

分工情况 Cooperation

组员 1 谢亦凡完成的部分：汇编代码的正确性验证、代码分析

组员 2 韩甜甜完成的部分：配置开发板环境、代码分析、代码性能测试、实验报告编写

组员 3 董晓冉完成的部分：配置开发板环境、代码分析、代码性能测试、实验报告编写

组员 4 朱科睿完成的部分：汇编代码编写、代码分析、代码性能测试

2025 年 5 月 22 日

摘要

本实验基于 RISC-V 架构，对 SM4 加密算法进行汇编层面的优化，并在设备上完成了性能测试。实验通过优化汇编代码逻辑（包括轮密钥预反转、寄存器高效利用、循环展开等策略），显著减少了加解密指令数量。在 Spike+PK 仿真环境下，加密指令数减少 8.912%，解密指令数减少 15.160%；在香橙派开发板实际测试中，加密性能提升约 11.489%，解密性能提升约 9.503%。此外，我们还尝试了基于 bitslicing 的并行优化方案。最终结果表明汇编优化对 SM4 算法性能实现了提升。

Abstract

Based on the RISC-V architecture, this experiment optimized the SM4 encryption algorithm at the assembly level and completed performance testing on the device. The experiment significantly reduced the number of encryption and decryption instructions by optimizing the assembly code logic (including round key pre-inversion, efficient use of registers, loop unrolling and other strategies). In the Spike+PK simulation environment, the number of encryption instructions decreased by 8.912%, and the number of decryption instructions decreased by 15.160%; in the actual test of the Orange Pi development board, the encryption performance increased by about 11.489%, and the decryption performance increased by about 9.053%. In addition, we also tried a parallel optimization solution based on bitslicing. The final results show that assembly optimization has improved the performance of the SM4 algorithm.

目录

分工情况 Cooperation	I
摘要 Abstract	I
目录 Table of Contents	II
1 软硬件环境	1
2 环境配置	1
3 汇编层面优化	2
3.1 代码实现	2
3.2 代码分析	2
3.3 Spike+PK 测试	4
3.3.1 测试结果	4
3.3.2 结果分析	4
3.4 香橙派开发板测试	5
3.4.1 测试结果	5
3.4.2 结果分析	7
3.5 Milk-V Meles 开发板测试	7
4 bitslicing 优化	7
5 总结 Summary	7
附录 Appendix	7

1 软硬件环境

1. 软件环境

- RISC-V 仿真器: Spike
- 操作系统: RISC-V
- 工具链依赖:
交叉编译器 `g++-riscv64-linux-gnu`、`gcc-riscv64-linux-gnu`;
依赖库 `libsctp-dev`;
构建工具 `cmake`、`make`、`git` 等

2. 硬件环境

- 香橙派开发板
- Milk-V Meles 开发板

2 环境配置

根据实验指导依次完成下述编译:

1. 安装依赖项
2. 编译 `libboundscheck` 源码
3. 编译 `openhitls` 源码

3 汇编层面优化

基于 RISC-V 架构和 openhitls 上的项目代码，我们编写相应的汇编指令来完成相应的优化：

3.1 代码实现

具体汇编代码详见附录。

3.2 代码分析

- 最小化内存访问

代码大量使用寄存器 t0-t6、a0-a7 进行中间计算和数据存储，很大程度上减少了内存访问；另外当前状态字始终保留在 a3-a6 中，XBOX 基地址仅加载一次并存储，实现了 s 盒查找的快速访问。

- 轮密钥预反转

将轮密钥进行预反转，使反转开销转移到预计算阶段，减少加解密过程中的开销。

- 向量指令

在 CBC 模式的实现中，将逐字节异或操作部分优化为使用 RISC-V 向量指令完成，通过 vle8.v 指令一次性加载 16 字节到向量寄存器，并利用 vxor.vv 实现向量级别的并行异或计算，最后通过 vse8.v 将结果存回内存。该优化显著减少了指令数量和内存访问次数，发挥了 RISC-V 向量扩展的并行计算优势，实现了高效的 CBC 块处理。

Listing 1: sm4.c

```
1 static inline void cbc_xor_block(uint8_t *block, const uint8_t *in1, const uint8_t *in2)
2 {
3     asm volatile (
4                                     // 设置向量寄存器配置：使用 e8 (8 位元素)，m1 组，向量长度 16
5         "li    t0, 16                \n\t" // 将立即数 16 (块大小) 加载到 t0
6         "vsetvli t1, t0, e8, m1     \n\t" // 根据 t0 设置向量长度，保证加载 16
           个 8 位元素
7
8         "vle8.v v0, (%1)             \n\t" // 从内存 in1 加载 16 个 8-bit 元素到
           v0
9         "vle8.v v1, (%2)             \n\t" // 从内存 in2 加载 16 个 8-bit 元素到
           v1
10
11        "vxor.vv v0, v0, v1           \n\t" // v0 = v0 XOR v1
12
13        "vse8.v v0, (%0)             \n\t" // 将 v0 中的数据存储在内存 block
14        :
```

```
15         : "r"(block), "r"(in1), "r"(in2)
16         : "t0", "t1", "v0", "v1", "memory"
17     );
18 }
```

- 循环展开

汇编代码中，我们采用循环展开 8 次的方式进行优化，通过该优化增加了 i-cache 的命中率。

Listing 2: sm4.S

```
1 SM4_ENCrypt_S:
2     PROLOGUE
3     Get_XBOX
4
5     li t6, 4
6 .L1:
7     ENC_ROUND_FUNCTION3 t0 t1 t2 t3
8     ENC_ROUND_FUNCTION3 t0 t1 t2 t3
9     addi t6, t6, -1
10    bgt t6, zero, .L1
11
12    EPILOGUE
```

- 指令细节

在进行乘 4 操作时，使用移位操作 slli，使运算效率更高。

- 值得注意的是

在实现 SM4 算法时,为了解决端序问题,并确保其计算逻辑与项目源代码中的 CRYPT_SM4_ROUND 函数保持严格一致:

首先，如果内存中原始输入数据 in[0:4] 为 0xaabbccdd，在 C 语言逻辑中将其作为 32 位字 X[0] 时可能被视为 0xddccbbaa，但在汇编寄存器它会直接存储为 0xaabbccdd。

然后我们采取如下操作：

在将轮密钥提供给汇编函数之前，每一个 32 位的轮密钥都必须进行一次 8 位字节的逆序处理（如果原始的轮密钥在逻辑上是 0xR0R1R2R3，那么在汇编函数实际加载和使用它时，必须确保其在寄存器中的表示是 0xR3R2R1R0）。通过这种处理，所使用的轮密钥与源代码字节序一致。

另一方面，XBOX 中存储的所有 S 盒查找数据同样需要进行逆序处理，此时在进行 $a7 = x1 \oplus x2 \oplus x3 \oplus rk$ 时，a7 就与 c 源码中 CRYPT_SM4_ROUND 中的 t 逆序。由于 $(XBOX_3)[((t) \gg 24) \& 0xff]$ 且 a7 是 t 的逆序，所以在汇编的 CRYPT_SM4_ROUND 中先取 a7 的低 8bit 查 XBOX3，其他以此类推。

3.3 Spike+PK 测试

3.3.1 测试结果

```
→ sm4_spike make ALG="sm4" CONF="-03 -DUSE_ASM=0" sw-build sw-run
make[1]: 进入目录"/home/skynion/experiment/CSAPP/openhitls/sm4_spike/src/sw"
make[1]: 离开目录"/home/skynion/experiment/CSAPP/openhitls/sm4_spike/src/sw"
make[1]: 进入目录"/home/skynion/experiment/CSAPP/openhitls/sm4_spike/src/sw"
bbl loader
Encrypt Cycle Count: 864
Decrypt Cycle Count: 1128
encrypt result: ec 27 ff 1b 6a ae 35 1a 01 75 d8 52 aa a1 53 d2
decrypt result: de ad be ef de ad be ef c0 ff ee aa c0 c0 fe fe
success
make[1]: 离开目录"/home/skynion/experiment/CSAPP/openhitls/sm4_spike/src/sw"
→ sm4_spike make ALG="sm4" CONF="-03 -DUSE_ASM=1" sw-build sw-run
make[1]: 进入目录"/home/skynion/experiment/CSAPP/openhitls/sm4_spike/src/sw"
make[1]: 离开目录"/home/skynion/experiment/CSAPP/openhitls/sm4_spike/src/sw"
make[1]: 进入目录"/home/skynion/experiment/CSAPP/openhitls/sm4_spike/src/sw"
bbl loader
Encrypt Cycle Count: 787
Decrypt Cycle Count: 957
encrypt result: ec 27 ff 1b 6a ae 35 1a 01 75 d8 52 aa a1 53 d2
decrypt result: de ad be ef de ad be ef c0 ff ee aa c0 c0 fe fe
success
make[1]: 离开目录"/home/skynion/experiment/CSAPP/openhitls/sm4_spike/src/sw"
```

图 1: Spike+PK 测试结果

3.3.2 结果分析

通过汇编语言层次的优化, 我们在 Spike+PK 环境上测试了解密指令数量, 显示加密指令数量减少了 8.912%; 解密指令数量减少了 15.160%, 实现了一定程度上的优化。

3.4 香橙派开发板测试

3.4.1 测试结果

```
orange@orange:~$ taskset -c 0 ./sm42
data size: 131072 byte
tick of 1s 24004466
turn on optimization
Encrypt: 3148ns
Decrypt: 3345ns
success
orange@orange:~$ taskset -c 0 ./sm42
data size: 131072 byte
tick of 1s 24004537
turn on optimization
Encrypt: 3099ns
Decrypt: 3333ns
success
orange@orange:~$ taskset -c 0 ./sm42
data size: 131072 byte
tick of 1s 24004456
turn on optimization
Encrypt: 3046ns
Decrypt: 3413ns
success
orange@orange:~$ taskset -c 0 ./sm42
data size: 131072 byte
tick of 1s 24004239
turn on optimization
Encrypt: 3056ns
Decrypt: 3465ns
success
orange@orange:~$ taskset -c 0 ./sm42
data size: 131072 byte
tick of 1s 24004059
turn on optimization
Encrypt: 3082ns
Decrypt: 3642ns
success
orange@orange:~$ taskset -c 0 ./sm42
data size: 131072 byte
tick of 1s 24004029
turn on optimization
Encrypt: 3049ns
Decrypt: 3392ns
success
orange@orange:~$ taskset -c 0 ./sm42
data size: 131072 byte
tick of 1s 24004025
turn on optimization
Encrypt: 3169ns
Decrypt: 3472ns
success
```

图 2: 测试结果 1


```

orange@orange:~$ taskset -c 0 ./sm42
data size: 131072 byte
tick of 1s 24003729
turn off optimization
Encrypt: 3488ns
Decrypt: 3768ns
success
orange@orange:~$ taskset -c 0 ./sm42
data size: 131072 byte
tick of 1s 24004122
turn off optimization
Encrypt: 3527ns
Decrypt: 3868ns
success
orange@orange:~$ taskset -c 0 ./sm42
data size: 131072 byte
tick of 1s 24004017
turn off optimization
Encrypt: 3489ns
Decrypt: 3849ns
success
orange@orange:~$ taskset -c 0 ./sm42
data size: 131072 byte
tick of 1s 24004426
turn off optimization
Encrypt: 3603ns
Decrypt: 3816ns
success
orange@orange:~$ taskset -c 0 ./sm42
data size: 131072 byte
tick of 1s 24004234
turn off optimization
Encrypt: 3495ns
Decrypt: 3726ns
success
orange@orange:~$ taskset -c 0 ./sm42
data size: 131072 byte
tick of 1s 24004370
turn off optimization
Encrypt: 3499ns
Decrypt: 3731ns
success
orange@orange:~$ taskset -c 0 ./sm42
data size: 131072 byte
tick of 1s 24004076
turn off optimization
Encrypt: 3395ns
Decrypt: 3707ns
success

```

图 3: 测试结果 2

测试	加密 (ns)		解密 (ns)	
	未优化	优化	未优化	优化
1	3488	3084	3768	3435
2	3527	3148	3868	3345
3	3489	3099	3849	3333
4	3603	3046	3816	3413
5	3495	3056	3726	3465
6	3499	3082	3731	3642
7	3395	3049	3707	3392
8	3438	3169	3798	3472

表 1: SM4 加密与解密在优化前后的执行时间

3.4.2 结果分析

通过汇编语言层次的优化,我们在香橙派开发板上测试了加解密运算的周期数,显示加密性能提升了 11.489%;解密性能提升了 9.503%,实现了一定程度上的优化。

3.5 Milk-V Meles 开发板测试

另外,我们还尝试在 Milk-V Meles 开发板上进行代码测试,由于环境配置较为复杂,测试未能成功实现。

4 bitslicing 优化

在本实验中,参考论文《How Fast Can SM4 be in Software?》,我们了解到一种使用 bitslicing 方法实现的 SM4 优化方案。该方法的核心思想是将 SM4 的处理单元从“块”变为“位”,然后利用 SIMD 指令并行处理多个数据块实现加速。

首先我们将这 n 个明文块中对应位置的比特重新打包到一组寄存器中,形成所谓的“slice”。例如在处理第 i 位时,某个 slice 中将包含所有明文块第 i 位的值。这可以使得每个逻辑指令同时作用于多个数据块,从而实现并行。

此时,SM4 轮函数中的轮密钥异或、S 盒非线性变换以及线性变换也均需修改为对 slice 进行操作。论文中提到,S 盒部分使用基于塔域分解和 Boyar 逻辑优化的方法,将查找表转化为逻辑电路,从而实现加速。线性变换通过位移和异或在 slice 间完成。轮函数的整体执行顺序保持不变,只是处理单位发生变化。

加密完成后,slice 数据需要按照刚刚所描述的相反的操作,从存储 slice 的寄存器中逐比特提取数据,然后将它们重新组合成 n 个密文数据块,得到正常的密文。解密过程同理。整个过程实现了对数据块的并行处理,极大提高了运行效率。

5 总结 Summary

本实验基于 RISC-V 架构,通过优化汇编代码,成功实现了 SM4 加解密算法的性能提升。在 Spike+PK 仿真环境下,加密指令数减少 8.9123%,解密指令数减少 15.160%;在香橙派开发板的测试中,加密效率提高 11.489%;解密效率提高 9.503%。此外,实验初步探索了基于 bitslicing 的并行优化方案。最终结果表明汇编优化对 SM4 算法性能实现了提升。

This experiment is based on the RISC-V architecture. By optimizing the assembly code, the performance of the SM4 encryption and decryption algorithm was successfully improved. In the Spike+PK simulation environment, the number of encryption instructions was reduced by 8.9123%, and the number of decryption instructions was reduced by 15.160%. In the test of the Orange Pi development board, the encryption efficiency was improved by 11.489%, and the decryption efficiency was improved by 9.503%. In addition, the experiment preliminarily explored the parallel optimization scheme based on bitslicing. The final results show that assembly optimization has improved the performance of the SM4 algorithm.

附录 Appendix

Listing 3: sm4.S

```
1 // a0 -> out
2 // a1 -> in
3 // a2 -> rk
4 // a3 -> XBOX_0
5 // a4 -> XBOX_1 = XBOX_0 + 1024
6 // a5 -> XBOX_2 = XBOX_0 + 2048
7 // a6 -> XBOX_3 = XBOX_0 + 3072
8 // a7 -> temp
9 // extern void SM4_Crypt_S(uint8_t *out, const uint8_t *in, const uint32_t *rk);
10 // t0 -> in[ 0: 4] = x0
11 // t1 -> in[ 4: 8] = x1
12 // t2 -> in[ 8:12] = x2
13 // t3 -> in[12:16] = x3
14
15 .macro PROLOGUE
16     // load input
17     lwu    t0,      0(a1)    // t0 -> in[ 0: 4]
18     lwu    t1,      4(a1)    // t1 -> in[ 4: 8]
19     lwu    t2,      8(a1)    // t2 -> in[ 8:12]
20     lwu    t3,     12(a1)    // t3 -> in[12:16]
21
22 .endm
23
24 .macro EPILOGUE
25     // save output
26     sw     t3,      0(a0)    // t3 -> out[ 0: 4]
27     sw     t2,      4(a0)    // t2 -> out[ 4: 8]
28     sw     t1,      8(a0)    // t1 -> out[ 8:12]
29     sw     t0,     12(a0)    // t0 -> out[12:16]
30
31     ret                                // return
32 .endm
33
34 .macro Get_XBOX
35     la     a3,    XBOX          // a3 -> XBOX_0
36     addi   a4,    a3,    1024    // a4 -> XBOX_1 = XBOX_0 + 1024
37     addi   a5,    a4,    1024    // a5 -> XBOX_2 = XBOX_0 + 2048
38     addi   a6,    a5,    1024    // a6 -> XBOX_3 = XBOX_0 + 3072
39 .endm
40
41 .macro CRYPT_SM4_ROUND x0, x1, x2, x3, rk_ofst
```

```

42    lwu    a7, \rk_ofst(a2)    // load rk 可以一次load 64bit rk 分两次用
43    xor    a7,    a7,    \x1
44    xor    a7,    a7,    \x2
45    xor    a7,    a7,    \x3    // a7 = x1 ^ x2 ^ x3 ^ rk
46
47    andi   t4,    a7,    0xFF    // t4 = a7 低 8bit
48    slli   t4,    t4,    2
49    add    t5,    t4,    a6    // a6 -> XBOX_3
50    lwu    t6,    0(t5)        // t4 = XBOX_3[t5(a7 & 0xff)]
51    xor    \x0,    \x0,    t6
52    srli   a7,    a7,    6
53
54    andi   t4,    a7,    0x3FC    // t4 = a7 & 0b1111111100
55    add    t5,    t4,    a5    // a5 -> XBOX_2
56    lwu    t6,    0(t5)        // t4 = XBOX_2[t5((a7 >> 8) & 0xff)]
57    xor    \x0,    \x0,    t6
58    srli   a7,    a7,    8
59
60    andi   t4,    a7,    0x3FC    // t4 = a7 低 8bit
61    add    t5,    t4,    a4    // a4 -> XBOX_1
62    lwu    t6,    0(t5)        // t4 = XBOX_1[t5((a7 >> 16) & 0xff)]
63    xor    \x0,    \x0,    t6
64    srli   a7,    a7,    8
65
66    andi   t4,    a7,    0x3FC    // t4 = a7 低 8bit
67    add    t5,    t4,    a3    // a3 -> XBOX_0
68    lwu    t6,    0(t5)        // t4 = XBOX_0[t5((a7 >> 24) & 0xff)]
69    xor    \x0,    \x0,    t6
70    .endm
71
72    .macro CRYPT_SM4_ROUND2 x0, x1, x2, x3, rk_ofst
73        ld    a7, \rk_ofst(a2)    // load rk
74        xor    t4,    \x1,    \x2
75        xor    a7,    a7,    \x3
76        xor    a7,    a7,    t4    // a7 = x1 ^ x2 ^ x3 ^ rk
77
78        andi   t4,    a7,    0xFF    // t4 = a7 低 8bit
79        slli   t4,    t4,    2
80        add    t5,    t4,    a6    // a6 -> XBOX_3
81        lwu    t6,    0(t5)        // t4 = XBOX_3[t5(a7 & 0xff)]
82        xor    \x0,    \x0,    t6
83        srli   a7,    a7,    6

```

```
84
85     andi t4, a7, 0x3FC // t4 = a7 & 0b1111111100
86     add t5, t4, a5 // a5 -> XBOX_2
87     lwu t6, 0(t5) // t4 = XBOX_2[t5((a7 >> 8) & 0xff)]
88     xor \x0, \x0, t6
89     srli a7, a7, 8
90
91     andi t4, a7, 0x3FC // t4 = a7 低 8bit
92     add t5, t4, a4 // a4 -> XBOX_1
93     lwu t6, 0(t5) // t4 = XBOX_1[t5((a7 >> 16) & 0xff)]
94     xor \x0, \x0, t6
95     srli a7, a7, 8
96
97     andi t4, a7, 0x3FC // t4 = a7 低 8bit
98     add t5, t4, a3 // a3 -> XBOX_0
99     lwu t6, 0(t5) // t4 = XBOX_0[t5((a7 >> 24) & 0xff)]
100    xor \x0, \x0, t6
101
102    srli a7, a7, 10
103    xor a7, a7, \x2
104    xor a7, a7, \x3
105    xor a7, a7, \x0 // a7 = x1 ^ x2 ^ x3 ^ rk
106
107    andi t4, a7, 0xFF // t4 = a7 低 8bit
108    slli t4, t4, 2
109    add t5, t4, a6 // a6 -> XBOX_3
110    lwu t6, 0(t5) // t4 = XBOX_3[t5(a7 & 0xff)]
111    xor \x1, \x1, t6
112    srli a7, a7, 6
113
114    andi t4, a7, 0x3FC // t4 = a7 & 0b1111111100
115    add t5, t4, a5 // a5 -> XBOX_2
116    lwu t6, 0(t5) // t4 = XBOX_2[t5((a7 >> 8) & 0xff)]
117    xor \x1, \x1, t6
118    srli a7, a7, 8
119
120    andi t4, a7, 0x3FC // t4 = a7 低 8bit
121    add t5, t4, a4 // a4 -> XBOX_1
122    lwu t6, 0(t5) // t4 = XBOX_1[t5((a7 >> 16) & 0xff)]
123    xor \x1, \x1, t6
124    srli a7, a7, 8
125
```

```

126     andi t4, a7, 0x3FC // t4 = a7 低 8bit
127     add t5, t4, a3 // a3 -> XBOX_0
128     lwu t6, 0(t5) // t4 = XBOX_0[t5((a7 >> 24) & 0xff)]
129     xor \x1, \x1, t6
130 .endm
131
132 .macro CRYPT_SM4_ROUND3 x0, x1, x2, x3
133     ld a7, 0(a2) // load rk
134     xor t4, \x1, \x2
135     xor a7, a7, \x3
136     xor a7, a7, t4 // a7 = x1 ^ x2 ^ x3 ^ rk
137
138     andi t4, a7, 0xFF // t4 = a7 低 8bit
139     slli t4, t4, 2
140     add t5, t4, a6 // a6 -> XBOX_3
141     lwu t5, 0(t5) // t4 = XBOX_3[t5(a7 & 0xff)]
142     srli a7, a7, 6
143     xor \x0, \x0, t5
144
145     andi t4, a7, 0x3FC // t4 = a7 & 0b1111111100
146     add t5, t4, a5 // a5 -> XBOX_2
147     lwu t5, 0(t5) // t4 = XBOX_2[t5((a7 >> 8) & 0xff)]
148     srli a7, a7, 8
149     xor \x0, \x0, t5
150
151     andi t4, a7, 0x3FC // t4 = a7 低 8bit
152     add t5, t4, a4 // a4 -> XBOX_1
153     lwu t5, 0(t5) // t4 = XBOX_1[t5((a7 >> 16) & 0xff)]
154     srli a7, a7, 8
155     xor \x0, \x0, t5
156
157     andi t4, a7, 0x3FC // t4 = a7 低 8bit
158     add t5, t4, a3 // a3 -> XBOX_0
159     lwu t5, 0(t5) // t4 = XBOX_0[t5((a7 >> 24) & 0xff)]
160     xor \x0, \x0, t5
161
162     srli a7, a7, 10
163     xor a7, a7, \x2
164     xor a7, a7, \x3
165     xor a7, a7, \x0 // a7 = x1 ^ x2 ^ x3 ^ rk
166
167     andi t4, a7, 0xFF // t4 = a7 低 8bit

```

```

168     slli  t4,    t4,    2
169     add  t5,    t4,    a6    // a6 -> XBOX_3
170     lwu   t5,    0(t5)      // t4 = XBOX_3[t5(a7 & 0xff)]
171     srli  a7,    a7,    6
172     xor   \x1,   \x1,    t5
173
174     andi  t4,    a7,    0x3FC // t4 = a7 & 0b1111111100
175     add  t5,    t4,    a5    // a5 -> XBOX_2
176     lwu   t5,    0(t5)      // t4 = XBOX_2[t5((a7 >> 8) & 0xff)]
177     srli  a7,    a7,    8
178     xor   \x1,   \x1,    t5
179
180     andi  t4,    a7,    0x3FC // t4 = a7 低 8bit
181     add  t5,    t4,    a4    // a4 -> XBOX_1
182     lwu   t5,    0(t5)      // t4 = XBOX_1[t5((a7 >> 16) & 0xff)]
183     srli  a7,    a7,    8
184     xor   \x1,   \x1,    t5
185
186     andi  t4,    a7,    0x3FC // t4 = a7 低 8bit
187     add  t5,    t4,    a3    // a3 -> XBOX_0
188     lwu   t5,    0(t5)      // t4 = XBOX_0[t5((a7 >> 24) & 0xff)]
189     xor   \x1,   \x1,    t5
190 .endm
191 .macro ENC_ROUND_FUNCTION x0, x1, x2, x3, rk_ofst
192     CRYPT_SM4_ROUND \x0, \x1, \x2, \x3, \rk_ofst
193     CRYPT_SM4_ROUND \x1, \x2, \x3, \x0, \rk_ofst + 4
194     CRYPT_SM4_ROUND \x2, \x3, \x0, \x1, \rk_ofst + 8
195     CRYPT_SM4_ROUND \x3, \x0, \x1, \x2, \rk_ofst + 12
196 .endm
197
198 .macro ENC_ROUND_FUNCTION2 x0, x1, x2, x3, rk_ofst
199     CRYPT_SM4_ROUND2 \x0, \x1, \x2, \x3, \rk_ofst
200     CRYPT_SM4_ROUND2 \x2, \x3, \x0, \x1, \rk_ofst + 8
201 .endm
202
203 .macro ENC_ROUND_FUNCTION3 x0, x1, x2, x3
204     CRYPT_SM4_ROUND3 \x0, \x1, \x2, \x3
205     addi  a2,    a2,    8
206     CRYPT_SM4_ROUND3 \x2, \x3, \x0, \x1
207     addi  a2,    a2,    8
208 .endm
209

```

```
210
211 .macro DEC_ROUND_FUNCTION x0, x1, x2, x3, rk_ofst
212     CRYPT_SM4_ROUND \x0, \x1, \x2, \x3, \rk_ofst + 12
213     CRYPT_SM4_ROUND \x1, \x2, \x3, \x0, \rk_ofst + 8
214     CRYPT_SM4_ROUND \x2, \x3, \x0, \x1, \rk_ofst + 4
215     CRYPT_SM4_ROUND \x3, \x0, \x1, \x2, \rk_ofst
216 .endm
217
218 .section .data
219
220 .align 3
221
222 XBOX:
223     // XBOX_0
224     .word 0x8e5b5bd5, 0xd0424292, 0x4da7a7ea, 0x06fbfbfd, 0xfc3333cf, 0x658787e2, 0xc
225         9f4f43d, 0x6bdedeb5
226     .word 0x4e585816, 0x6edadab4, 0x44505014, 0xca0b0bc1, 0x88a0a028, 0x17efeff8, 0x9
227         cb0b02c, 0x11141405
228     .word 0x87acac2b, 0xfb9d9d66, 0xf26a6a98, 0xaed9d977, 0x82a8a82a, 0x46fafabc, 0x
229         14101004, 0xcf0f0fc0
230     .word 0x02aaaaa8, 0x54111145, 0x5f4c4c13, 0xbe989826, 0x6d252548, 0x9e1a1a84, 0x1
231         e181806, 0xfd66669b
232     .word 0xec72729e, 0x4a090943, 0x10414151, 0x24d3d3f7, 0xd5464693, 0x53bfbfec, 0xf
233         862629a, 0x92e9e97b
234     .word 0xffcccc33, 0x04515155, 0x272c2c0b, 0x4f0d0d42, 0x59b7b7ee, 0xf33f3fcc, 0x1
235         cb2b2ae, 0xea898963
236     .word 0x749393e7, 0x7fceceb1, 0x6c70701c, 0x0da6a6ab, 0xed2727ca, 0x28202008, 0x
237         48a3a3eb, 0xc1565697
238     .word 0x80020282, 0xa37f7fdc, 0xc4525296, 0x12ebefb9, 0xa1d5d574, 0xb33e3e8d, 0xc
239         3fcfc3f, 0x3e9a9aa4
240     .word 0x5b1d1d46, 0x1b1c1c07, 0x3b9e9ea5, 0x0cf3f3ff, 0x3fcfcff0, 0xbfcdcd72, 0x4
241         b5c5c17, 0x52eaeab8
242     .word 0x8f0e0e81, 0x3d656558, 0xccf0f03c, 0x7d646419, 0x7e9b9be5, 0x91161687, 0x
243         733d3d4e, 0x08a2a2aa
244     .word 0xc8a1a169, 0xc7adad6a, 0x85060683, 0x7acacab0, 0xb5c5c570, 0xf4919165, 0xb
245         26b6bd9, 0xa72e2e89
246     .word 0x18e3e3fb, 0x47afafe8, 0x333c3c0f, 0x672d2d4a, 0xb0c1c171, 0xe595957, 0xe
247         976769f, 0xe1d4d435
248     .word 0x6678781e, 0xb4909024, 0x3638380e, 0x2679795f, 0xef8d8d62, 0x38616159, 0x
249         954747d2, 0x2a8a8aa0
250     .word 0xb1949425, 0xaa888822, 0x8cf1f17d, 0xd7ecec3b, 0x05040401, 0xa5848421, 0x
251         98e1e179, 0x9b1e1e85
```



```
238 .word 0x845353d7, 0x00000000, 0x5e191947, 0x0b5d5d56, 0xe37e7e9d, 0x9f4f4fd0, 0
    xbb9c9c27, 0x1a494953
239 .word 0x7c31314d, 0xead8d836, 0x0a080802, 0x7b9f9fe4, 0x208282a2, 0xd41313c7, 0xe
    82323cb, 0xe67a7a9c
240 .word 0x42ababe9, 0x43fefebd, 0xa22a2a88, 0x9a4b4bd1, 0x40010141, 0xdb1f1fc4, 0xd
    8e0e038, 0x61d6d6b7
241 .word 0x2f8e8ea1, 0x2bdfdf4, 0x3acbcbf1, 0xf63b3bcd, 0x1de7e7fa, 0xe5858560, 0x
    41545415, 0x258686a3
242 .word 0x608383e3, 0x16babaac, 0x2975755c, 0x349292a6, 0xf76e6e99, 0xe4d0d034, 0x
    7268681a, 0x01555554
243 .word 0x19b6b6af, 0xdf4e4e91, 0xfac8c832, 0xf0c0c030, 0x21d7d7f6, 0xbc32328e, 0x
    75c6c6b3, 0x6f8f8fe0
244 .word 0x6974741d, 0x2eddbdf5, 0x6a8b8be1, 0x96b8b82e, 0x8a0a0a80, 0xfe999967, 0xe
    22b2bc9, 0xe0818161
245 .word 0xc00303c3, 0x8da4a429, 0xaf8c8c23, 0x07aeaea9, 0x3934340d, 0x1f4d4d52, 0x
    7639394f, 0xd3bdbd6e
246 .word 0x815757d6, 0xb76f6fd8, 0xebdc37, 0x51151544, 0xa67b7bdd, 0x09f7f7fe, 0xb
    63a3a8c, 0x93bcbcb2f
247 .word 0xf0c0c03, 0x03fffffc, 0xc2a9a96b, 0xbac9c973, 0xd9b5b56c, 0xdc1b1b6d, 0x
    376d6d5a, 0x15454550
248 .word 0xb936368f, 0x776c6c1b, 0x13bebead, 0xda4a4a90, 0x57eeeb9, 0xa97777de, 0x4
    cf2f2be, 0x83fdfd7e
249 .word 0x55444411, 0xbd6767da, 0x2c71715d, 0x45050540, 0x637c7c1f, 0x50404010, 0x
    3269695b, 0xb86363db
250 .word 0x2228280a, 0xc50707c2, 0xf5c4c431, 0xa822228a, 0x319696a7, 0xf93737ce, 0x
    97eded7a, 0x49f6f6bf
251 .word 0x99b4b42d, 0xa4d1d175, 0x904343d3, 0x5a484812, 0x58e2e2ba, 0x719797e6, 0x
    64d2d2b6, 0x70c2c2b2
252 .word 0xad26268b, 0xcda5a568, 0xcb5e5e95, 0x6229294b, 0x3c30300c, 0xce5a5a94, 0
    xabddd76, 0x86f9f97f
253 .word 0xf1959564, 0x5de6e6bb, 0x35c7c7f2, 0x2d242409, 0xd11717c6, 0xd6b9b96f, 0
    xde1b1bc5, 0x94121286
254 .word 0x78606018, 0x30c3c3f3, 0x89f5f57c, 0x5cb3b3ef, 0xd2e8e83a, 0xac7373df, 0x
    7935354c, 0xa0808020
255 .word 0x9de5e578, 0x56bbbbbbed, 0x237d7d5e, 0xc6f8f83e, 0x8b5f5fd4, 0xe72f2fc8, 0
    xdde4e439, 0x68212149
256
257 // XBOX_1
258 .word 0xd58e5b5b, 0x92d04242, 0xea4da7a7, 0xfd06fbfb, 0xcffc3333, 0xe2658787, 0x3
    dc9f4f4, 0xb56bdede
259 .word 0x164e5858, 0xb46edada, 0x14445050, 0xc1ca0b0b, 0x2888a0a0, 0xf817efef, 0x2
    c9cb0b0, 0x05111414
```

```
260 .word 0x2b87acac, 0x66fb9d9d, 0x98f26a6a, 0x77aed9d9, 0x2a82a8a8, 0xbc46fafa, 0x
    04141010, 0xc0cf0f0f
261 .word 0xa802aaaa, 0x45541111, 0x135f4c4c, 0x26be9898, 0x486d2525, 0x849e1a1a, 0x
    061e1818, 0x9bfd6666
262 .word 0x9eec7272, 0x434a0909, 0x51104141, 0xf724d3d3, 0x93d54646, 0xec53bfbf, 0x9
    af86262, 0x7b92e9e9
263 .word 0x33ffcccc, 0x55045151, 0x0b272c2c, 0x424f0d0d, 0xee59b7b7, 0xccf33f3f, 0
    xae1cb2b2, 0x63ea8989
264 .word 0xe7749393, 0xb17fcece, 0x1c6c7070, 0xab0da6a6, 0xcaed2727, 0x08282020, 0
    xeb48a3a3, 0x97c15656
265 .word 0x82800202, 0xdca37f7f, 0x96c45252, 0xf912ebef, 0x74a1d5d5, 0x8db33e3e, 0x3
    fc3fcfc, 0xa43e9a9a
266 .word 0x465b1d1d, 0x071b1c1c, 0xa53b9e9e, 0xff0cf3f3, 0xf03fcfcf, 0x72bfcdcd, 0x
    174b5c5c, 0xb852eaea
267 .word 0x818f0e0e, 0x583d6565, 0x3cccf0f0, 0x197d6464, 0xe57e9b9b, 0x87911616, 0x4
    e733d3d, 0xaa08a2a2
268 .word 0x69c8a1a1, 0x6ac7adad, 0x83850606, 0xb07acaca, 0x70b5c5c5, 0x65f49191, 0xd
    9b26b6b, 0x89a72e2e
269 .word 0xf1b8e3e3, 0xe847afaf, 0x0f333c3c, 0x4a672d2d, 0x71b0c1c1, 0x570e5959, 0x9
    fe97676, 0x35e1d4d4
270 .word 0x1e667878, 0x24b49090, 0x0e363838, 0x5f267979, 0x62ef8d8d, 0x59386161, 0xd
    2954747, 0xa02a8a8a
271 .word 0x25b19494, 0x22aa8888, 0x7d8cf1f1, 0x3bd7ecec, 0x01050404, 0x21a58484, 0x
    7998e1e1, 0x859b1e1e
272 .word 0xd7845353, 0x00000000, 0x475e1919, 0x560b5d5d, 0x9de37e7e, 0xd09f4f4f, 0x
    27bb9c9c, 0x531a4949
273 .word 0x4d7c3131, 0x36eed8d8, 0x020a0808, 0xe47b9f9f, 0xa2208282, 0xc7d41313, 0
    xcbe82323, 0x9ce67a7a
274 .word 0xe942abab, 0xbd43fefe, 0x88a22a2a, 0xd19a4b4b, 0x41400101, 0xc4db1f1f, 0x
    38d8e0e0, 0xb761d6d6
275 .word 0xa12f8e8e, 0xf42bdfdf, 0xf13acbc, 0xcdf63b3b, 0xfa1de7e7, 0x60e58585, 0x
    15415454, 0xa3258686
276 .word 0xe3608383, 0xac16baba, 0x5c297575, 0xa6349292, 0x99f76e6e, 0x34e4d0d0, 0x1
    a726868, 0x54015555
277 .word 0xaf19b6b6, 0x91df4e4e, 0x32fac8c8, 0x30f0c0c0, 0xf621d7d7, 0x8ebc3232, 0xb
    375c6c6, 0xe06f8f8f
278 .word 0x1d697474, 0xf52edbdb, 0xe16a8b8b, 0x2e96b8b8, 0x808a0a0a, 0x67fe9999, 0xc
    9e22b2b, 0x61e08181
279 .word 0xc3c00303, 0x298da4a4, 0x23af8c8c, 0xa907aeae, 0x0d393434, 0x521f4d4d, 0x4
    f763939, 0x6ed3dbdb
280 .word 0xd6815757, 0xd8b76f6f, 0x37ebdc, 0x44511515, 0xdda67b7b, 0xfe09f7f7, 0x8
    cb63a3a, 0x2f93bcb
```

```
281 .word 0x030f0c0c, 0xfc03ffff, 0x6bc2a9a9, 0x73bac9c9, 0x6cd9b5b5, 0x6ddcb1b1, 0x5
    a376d6d, 0x50154545
282 .word 0x8fb93636, 0x1b776c6c, 0xad13bebe, 0x90da4a4a, 0xb957eeee, 0xdea97777, 0
    xbe4cf2f2, 0x7e83fdfd
283 .word 0x11554444, 0xdabd6767, 0x5d2c7171, 0x40450505, 0x1f637c7c, 0x10504040, 0x5
    b326969, 0xdbb86363
284 .word 0x0a222828, 0xc2c50707, 0x31f5c4c4, 0x8aa82222, 0xa7319696, 0xcef93737, 0x7
    a97eded, 0xbf49f6f6
285 .word 0x2d99b4b4, 0x75a4d1d1, 0xd3904343, 0x125a4848, 0xba58e2e2, 0xe6719797, 0xb
    664d2d2, 0xb270c2c2
286 .word 0x8bad2626, 0x68cda5a5, 0x95cb5e5e, 0x4b622929, 0x0c3c3030, 0x94ce5a5a, 0x
    76abdddd, 0xf86f9f9
287 .word 0x64f19595, 0xbb5de6e6, 0xf235c7c7, 0x092d2424, 0xc6d11717, 0x6fd6b9b9, 0xc
    5de1b1b, 0x86941212
288 .word 0x18786060, 0xf330c3c3, 0x7c89f5f5, 0xef5cb3b3, 0x3ad2e8e8, 0xdfac7373, 0x4
    c793535, 0x20a08080
289 .word 0x789de5e5, 0xed56bbbb, 0x5e237d7d, 0x3ec6f8f8, 0xd48b5f5f, 0xc8e72f2f, 0x
    39dde4e4, 0x49682121
290
291 // XBOX_2
292 .word 0x5bd58e5b, 0x4292d042, 0xa7ea4da7, 0xfbfd06fb, 0x33cffc33, 0x87e26587, 0xf
    43dc9f4, 0xdeb56bde
293 .word 0x58164e58, 0xdab46eda, 0x50144450, 0x0bc1ca0b, 0xa02888a0, 0xeff817ef, 0xb
    02c9cb0, 0x14051114
294 .word 0xac2b87ac, 0x9d66fb9d, 0x6a98f26a, 0xd977aed9, 0xa82a82a8, 0xfabc46fa, 0x
    10041410, 0x0fc0cf0f
295 .word 0xaa802aa, 0x11455411, 0x4c135f4c, 0x9826be98, 0x25486d25, 0x1a849e1a, 0x
    18061e18, 0x669bfd66
296 .word 0x729eec72, 0x09434a09, 0x41511041, 0xd3f724d3, 0x4693d546, 0xbfec53bf, 0x
    629af862, 0xe97b92e9
297 .word 0xcc33ffcc, 0x51550451, 0x2c0b272c, 0xd424f0d, 0xb7ee59b7, 0x3fccf33f, 0xb
    2ae1cb2, 0x8963ea89
298 .word 0x93e77493, 0xceb17fce, 0x701c6c70, 0xa6ab0da6, 0x27caed27, 0x20082820, 0xa
    3eb48a3, 0x5697c156
299 .word 0x02828002, 0x7fdca37f, 0x5296c452, 0xebf912eb, 0xd574a1d5, 0x3e8db33e, 0
    xfc3fc3fc, 0x9aa43e9a
300 .word 0x1d465b1d, 0x1c071b1c, 0x9ea53b9e, 0xf3ff0cf3, 0xcff03fcf, 0xcd72bfcd, 0x5
    c174b5c, 0xeab852ea
301 .word 0x0e818f0e, 0x65583d65, 0xf03cccf0, 0x64197d64, 0x9be57e9b, 0x16879116, 0x3
    d4e733d, 0xa2aa08a2
302 .word 0xa169c8a1, 0xad6ac7ad, 0x06838506, 0xcab07aca, 0xc570b5c5, 0x9165f491, 0x6
    bd9b26b, 0x2e89a72e
```

```
303 .word 0xe3fb18e3, 0xafe847af, 0x3c0f333c, 0x2d4a672d, 0xc171b0c1, 0x59570e59, 0x
    769fe976, 0xd435e1d4
304 .word 0x781e6678, 0x9024b490, 0x380e3638, 0x795f2679, 0x8d62ef8d, 0x61593861, 0x
    47d29547, 0x8aa02a8a
305 .word 0x9425b194, 0x8822aa88, 0xf17d8cf1, 0xec3bd7ec, 0x04010504, 0x8421a584, 0xe
    17998e1, 0x1e859b1e
306 .word 0x53d78453, 0x00000000, 0x19475e19, 0x5d560b5d, 0x7e9de37e, 0x4fd09f4f, 0x9
    c27bb9c, 0x49531a49
307 .word 0x314d7c31, 0xd836eed8, 0x08020a08, 0x9fe47b9f, 0x82a22082, 0x13c7d413, 0x
    23cbe823, 0x7a9ce67a
308 .word 0xab942ab, 0xfebd43fe, 0x2a88a22a, 0x4bd19a4b, 0x01414001, 0x1fc4db1f, 0xe
    038d8e0, 0xd6b761d6
309 .word 0x8ea12f8e, 0xdff42bdf, 0xcbf13acb, 0x3bcd6f63b, 0xe7fa1de7, 0x8560e585, 0x
    54154154, 0x86a32586
310 .word 0x83e36083, 0xbaac16ba, 0x755c2975, 0x92a63492, 0x6e99f76e, 0xd034e4d0, 0x
    681a7268, 0x55540155
311 .word 0xb6af19b6, 0x4e91df4e, 0xc832fac8, 0xc030f0c0, 0xd7f621d7, 0x328ebc32, 0xc
    6b375c6, 0x8fe06f8f
312 .word 0x741d6974, 0xdbf52edb, 0x8be16a8b, 0xb82e96b8, 0x0a808a0a, 0x9967fe99, 0x2
    bc9e22b, 0x8161e081
313 .word 0x03c3c003, 0xa4298da4, 0x8c23af8c, 0xaea907ae, 0x340d3934, 0x4d521f4d, 0x
    394f7639, 0xbd6ed3bd
314 .word 0x57d68157, 0x6fd8b76f, 0xdc37ebdc, 0x15445115, 0x7bdda67b, 0xf7fe09f7, 0x3
    a8cb63a, 0xbc2f93bc
315 .word 0x0c030f0c, 0xffffc03ff, 0xa96bc2a9, 0xc973bac9, 0xb56cd9b5, 0xb16ddcb1, 0x6
    d5a376d, 0x45501545
316 .word 0x368fb936, 0x6c1b776c, 0xbead13be, 0x4a90da4a, 0xeeb957ee, 0x77dea977, 0xf
    2be4cf2, 0xfd7e83fd
317 .word 0x44115544, 0x67dabd67, 0x715d2c71, 0x05404505, 0x7c1f637c, 0x40105040, 0x
    695b3269, 0x63dbb863
318 .word 0x280a2228, 0x07c2c507, 0xc431f5c4, 0x228aa822, 0x96a73196, 0x37cef937, 0
    xed7a97ed, 0xf6bf49f6
319 .word 0xb42d99b4, 0xd175a4d1, 0x43d39043, 0x48125a48, 0xe2ba58e2, 0x97e67197, 0xd
    2b664d2, 0xc2b270c2
320 .word 0x268bad26, 0xa568cda5, 0x5e95cb5e, 0x294b6229, 0x300c3c30, 0x5a94ce5a, 0
    xdd76abdd, 0xf97f86f9
321 .word 0x9564f195, 0xe6bb5de6, 0xc7f235c7, 0x24092d24, 0x17c6d117, 0xb96fd6b9, 0x1
    bc5de1b, 0x12869412
322 .word 0x60187860, 0xc3f330c3, 0xf57c89f5, 0xb3ef5cb3, 0xe83ad2e8, 0x73dfac73, 0x
    354c7935, 0x8020a080
323 .word 0xe5789de5, 0xbbed56bb, 0x7d5e237d, 0xf83ec6f8, 0x5fd48b5f, 0x2fc8e72f, 0xe
    439dde4, 0x21496821
```

```
324
325 // XBOX_3
326 .word 0x5b5bd58e, 0x424292d0, 0xa7a7ea4d, 0xfbfbfd06, 0x3333cffc, 0x8787e265, 0xf
    4f43dc9, 0xdedeb56b
327 .word 0x5858164e, 0xdadab46e, 0x50501444, 0x0b0bc1ca, 0xa0a02888, 0xefeff817, 0xb
    0b02c9c, 0x14140511
328 .word 0xacac2b87, 0x9d9d66fb, 0x6a6a98f2, 0xd9d977ae, 0xa8a82a82, 0xfafabc46, 0x
    10100414, 0x0f0fc0cf
329 .word 0xaaaaa802, 0x11114554, 0x4c4c135f, 0x989826be, 0x2525486d, 0x1a1a849e, 0x
    1818061e, 0x66669bfd
330 .word 0x72729eec, 0x0909434a, 0x41415110, 0xd3d3f724, 0x464693d5, 0xbfbfec53, 0x
    62629af8, 0xe9e97b92
331 .word 0xcccc33ff, 0x51515504, 0x2c2c0b27, 0xd0d0424f, 0xb7b7ee59, 0x3f3fccf3, 0xb
    2b2ae1c, 0x898963ea
332 .word 0x9393e774, 0xceceb17f, 0x70701c6c, 0xa6a6ab0d, 0x2727caed, 0x20200828, 0xa
    3a3eb48, 0x565697c1
333 .word 0x02028280, 0x7f7fdca3, 0x525296c4, 0xebebf912, 0xd5d574a1, 0x3e3e8db3, 0
    xfcfc3fc3, 0x9a9aa43e
334 .word 0x1d1d465b, 0x1c1c071b, 0x9e9ea53b, 0xf3f3ff0c, 0xcfcff03f, 0xcdcd72bf, 0x5
    c5c174b, 0xaeaeb852
335 .word 0x0e0e818f, 0x6565583d, 0xf0f03ccc, 0x6464197d, 0x9b9be57e, 0x16168791, 0x3
    d3d4e73, 0xa2a2aa08
336 .word 0xa1a169c8, 0xadad6ac7, 0x06068385, 0xcacab07a, 0xc5c570b5, 0x919165f4, 0x6
    b6bd9b2, 0x2e2e89a7
337 .word 0xe3e3fb18, 0xafafe847, 0x3c3c0f33, 0x2d2d4a67, 0xc1c171b0, 0x5959570e, 0x
    76769fe9, 0xd4d435e1
338 .word 0x78781e66, 0x909024b4, 0x38380e36, 0x79795f26, 0x8d8d62ef, 0x61615938, 0x
    4747d295, 0x8a8aa02a
339 .word 0x949425b1, 0x888822aa, 0xf1f17d8c, 0xeccec3bd7, 0x04040105, 0x848421a5, 0xe
    1e17998, 0x1e1e859b
340 .word 0x5353d784, 0x00000000, 0x1919475e, 0x5d5d560b, 0x7e7e9de3, 0x4f4fd09f, 0x9
    c9c27bb, 0x4949531a
341 .word 0x31314d7c, 0xd8d836ee, 0x0808020a, 0x9f9fe47b, 0x8282a220, 0x1313c7d4, 0x
    2323cbe8, 0x7a7a9ce6
342 .word 0xababe942, 0xfefebd43, 0x2a2a88a2, 0x4b4bd19a, 0x01014140, 0x1f1fc4db, 0xe
    0e038d8, 0xd6d6b761
343 .word 0x8e8ea12f, 0xdfdff42b, 0xcbcbf13a, 0x3b3bcd6f, 0xe7e7fa1d, 0x858560e5, 0x
    54541541, 0x8686a325
344 .word 0x8383e360, 0xbabaaac16, 0x75755c29, 0x9292a634, 0x6e6e99f7, 0xd0d034e4, 0x
    68681a72, 0x55555401
345 .word 0xb6b6af19, 0x4e4e91df, 0xc8c832fa, 0xc0c030f0, 0xd7d7f621, 0x32328ebc, 0xc
    6c6b375, 0x8f8fe06f
```

```
346 .word 0x74741d69, 0xdbdbf52e, 0x8b8be16a, 0xb8b82e96, 0x0a0a808a, 0x999967fe, 0x2
    b2bc9e2, 0x818161e0
347 .word 0x0303c3c0, 0xa4a4298d, 0x8c8c23af, 0xaeaea907, 0x34340d39, 0x4d4d521f, 0x
    39394f76, 0xbdbd6ed3
348 .word 0x5757d681, 0x6f6fd8b7, 0xdcdc37eb, 0x15154451, 0x7b7bdda6, 0xf7f7fe09, 0x3
    a3a8cb6, 0xbcbc2f93
349 .word 0x0c0c030f, 0xfffffc03, 0xa9a96bc2, 0xc9c973ba, 0xb5b56cd9, 0xb1b16ddc, 0x6
    d6d5a37, 0x45455015
350 .word 0x36368fb9, 0x6c6c1b77, 0xbebead13, 0x4a4a90da, 0xeeeeb957, 0x7777dea9, 0xf
    2f2be4c, 0xfdfd7e83
351 .word 0x44441155, 0x6767dabd, 0x71715d2c, 0x05054045, 0x7c7c1f63, 0x40401050, 0x
    69695b32, 0x6363dbb8
352 .word 0x28280a22, 0x0707c2c5, 0xc4c431f5, 0x22228aa8, 0x9696a731, 0x3737cef9, 0
    xeded7a97, 0xf6f6bf49
353 .word 0xb4b42d99, 0xd1d175a4, 0x4343d390, 0x4848125a, 0xe2e2ba58, 0x9797e671, 0xd
    2d2b664, 0xc2c2b270
354 .word 0x26268bad, 0xa5a568cd, 0x5e5e95cb, 0x29294b62, 0x30300c3c, 0x5a5a94ce, 0
    xdddd76ab, 0xf9f97f86
355 .word 0x959564f1, 0xe6e6bb5d, 0xc7c7f235, 0x2424092d, 0x1717c6d1, 0xb9b96fd6, 0x1
    b1bc5de, 0x12128694
356 .word 0x60601878, 0xc3c3f330, 0xf5f57c89, 0xb3b3ef5c, 0xe8e83ad2, 0x7373dfac, 0x
    35354c79, 0x808020a0
357 .word 0xe5e5789d, 0xbbbbbed56, 0x7d7d5e23, 0xf8f83ec6, 0x5f5fd48b, 0x2f2fc8e7, 0xe
    4e439dd, 0x21214968
358
359 .section .text
360
361 .global SM4_ENCrypt_S, SM4_DECrypt_S
362
363 SM4_ENCrypt_S:
364     PROLOGUE
365     Get_XBOX
366
367     li t6, 4
368 .L1:
369     ENC_ROUND_FUNCTION3 t0 t1 t2 t3
370     ENC_ROUND_FUNCTION3 t0 t1 t2 t3
371     addi t6, t6, -1
372     bgt t6, zero, .L1
373
374
375     EPILOGUE
```

```
376
377 SM4_DECrypt_S:
378     PROLOGUE
379     Get_XBOX
380
381     DEC_ROUND_FUNCTION t0 t1 t2 t3 112
382     DEC_ROUND_FUNCTION t0 t1 t2 t3 96
383     DEC_ROUND_FUNCTION t0 t1 t2 t3 80
384     DEC_ROUND_FUNCTION t0 t1 t2 t3 64
385     DEC_ROUND_FUNCTION t0 t1 t2 t3 48
386     DEC_ROUND_FUNCTION t0 t1 t2 t3 32
387     DEC_ROUND_FUNCTION t0 t1 t2 t3 16
388     DEC_ROUND_FUNCTION t0 t1 t2 t3 0
389
390     EPILOGUE
```

参考文献

- [1] Xin Miao, Chun Guo, Meiqin Wang, and Weijia Wang. 2022. How Fast Can SM4 be innbsp;Software? In Information Security and Cryptology: 18th International Conference, In-scrypt 2022, Beijing, China, December 11–13, 2022, Revised Selected Papers. Springer-Verlag, Berlin, Heidelberg, 3–22. https://doi.org/10.1007/978-3-031-26553-2_1