РусКрипто

Расширение системы команд RISC-V для ГОСТ-криптографии

Матюков Андрей,
Альянс RISC-V

XXVII НАУЧНО-ПРАКТИЧЕСКАЯ КОНФЕРЕНЦИЯ

Рассматриваемые особенности КФС

- Ограниченные ресурсы
- Обработка данных в реальном времени
- Высокая связность (connectivity) с другими элементами
- Должны обеспечивать устойчивость к атакам (в т.ч. side-channel)

РусКрипто

Рассматриваемые особенности КФС

- Ограниченные ресурсы
- Обработка данных в реальном времени
- Высокая связность (connectivity) с другими элементами
- Должны обеспечивать устойчивость к атакам (в т.ч. side-channel)

⇒ Влияет на выбор алгоритмов, которые используются в КФС
- ⇒ Должны обеспечивать достаточную производительность, чтобы уместиться в заданные рамки обработки кадров
  - ⇒ Низкоресурсные
- ⇒ Обеспечивать требуемый уровень безопасности

РусКрипто

Варианты ускорения

- Алгоритмическая оптимизация
- Применение SIMD / Vector инструкций
- Специализированный набор инструкций в составе CPU-ядра
- Криптографические со-процессоры, ускорители

РусКрипто

Обзор наборов инструкций CPU для некоторых алгоритмов криптографии

РусКрипто

	X86	ARMv8	RISC-V
AES	(V)AES-NI (up to 512-bit SIMD) (V)PCLMUL (up to 512-bit SIMD)	Crypto Ext • aes{e,d} • pmull	Zkned (Scalar) Zvkned (Vector)
SHA2	SHA-NI (up to 256-bit SIMD) • Sha256 • Sha512	Crypto Ext • Sha256 • Sha512	Zknh (Scalar) Zvknh (Vector)

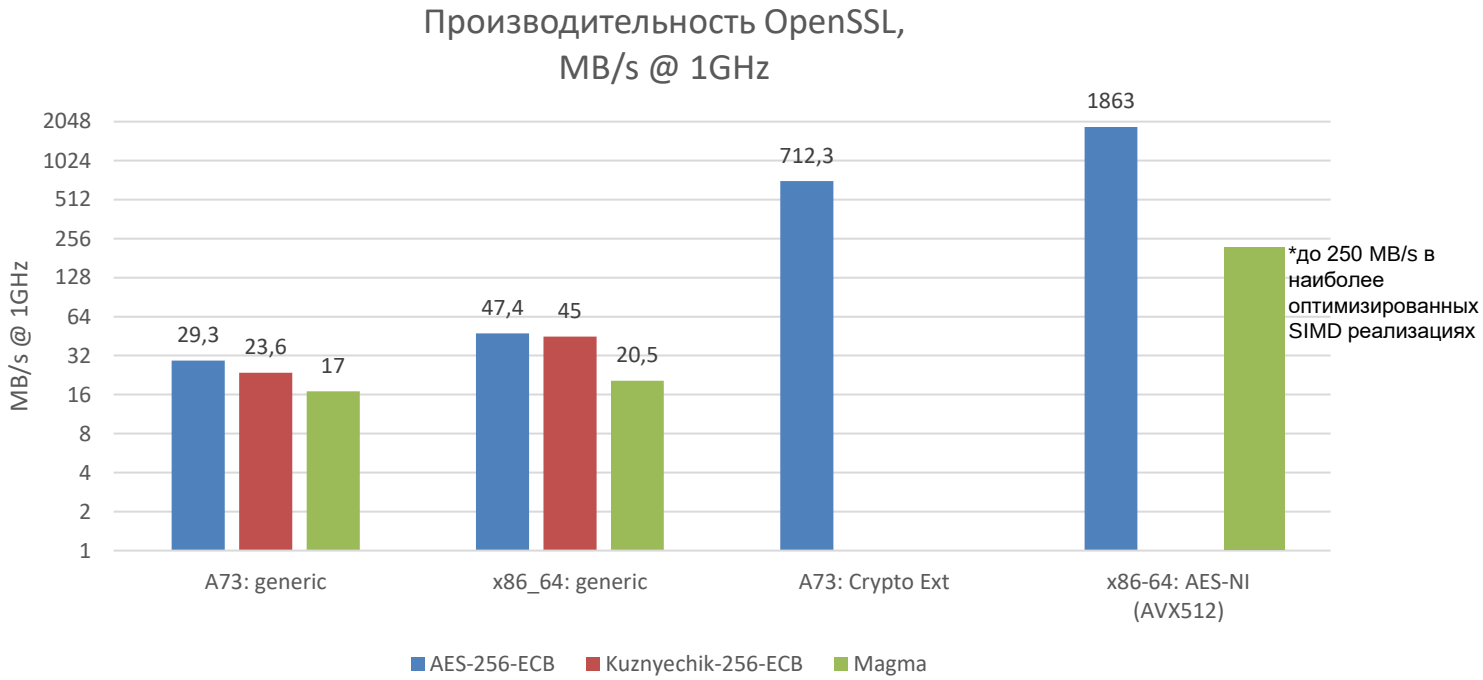
Обзор наборов инструкций CPU для некоторых алгоритмов криптографии

РусКрипто

	X86	ARMv8	RISC-V
AES	(V)AES-NI (up to 512-bit SIMD) (V)PCLMUL (up to 512-bit SIMD)	Crypto Ext • aes{e,d} • pmull	Zkned (Scalar) Zvkned (Vector)
SHA2	SHA-NI (up to 256-bit SIMD) • Sha256 • Sha512	Crypto Ext • Sha256 • Sha512	Zknh (Scalar) Zvknh (Vector)
Кузнечик	-	-	**Возможно**
Магма	-	-	**Возможно**
Стрибог	-	-	**Возможно**

Производительность крипто-ISA

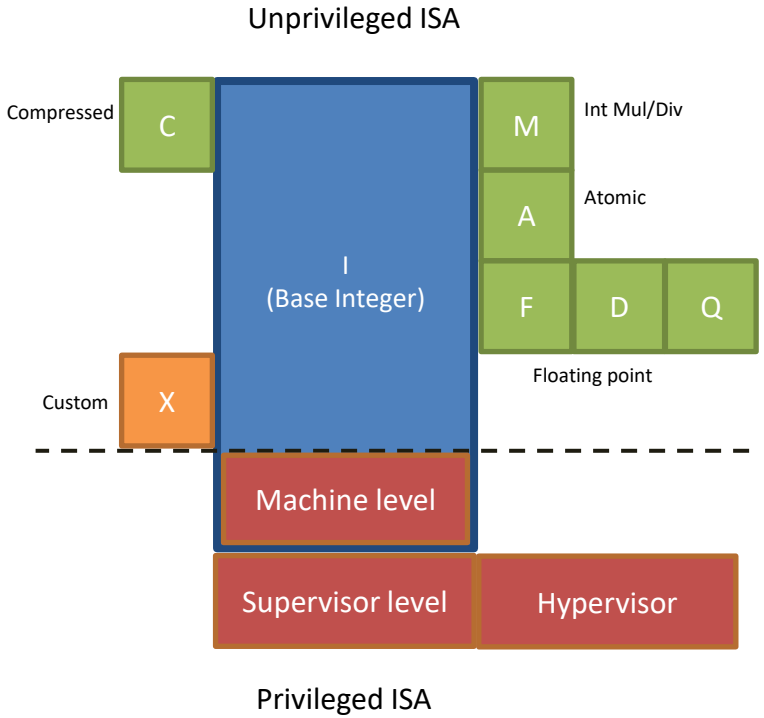
РусКрипто



Почему RISC-V?

- Свободная и открытая процессорная архитектура
- Модульная и расширяемая система команд
- Подходит для широкого спектра оборудования (от smart-карт до HPC)
- Заложены механизмы создания пользовательских (custom) расширений

РусКрипто



Отечественные микроконтроллеры RISC-V

РусКрипто

	Ядро	ISA
К1986ВК025 (Микрон)	BM310S	RV32IMC
MDR1206AFI (Миландр)	BM310S0	RV32IMCN + Bitmanip + Scalar Crypto + Xgost
К1948ВК018 MIK32 «Амур» (Микрон)	SCR1	RV32IMC
К1921ВГ015 (НИИЭТ)	BM310S6	RV32IMFCN + Bitmanip

Потребность в стандартизации

РусКрипто

- Расширение номенклатуры RISC-V IP с поддержкой ГОСТ-расширения от разных вендоров
- Унификация поддержки в экосистеме ПО (компиляторы, симуляторы, библиотеки)
- Упрощение поддержки в СКЗИ

Потребность в стандартизации

РусКрипто

- Расширение номенклатуры RISC-V IP с поддержкой ГОСТ-расширения от разных вендоров
- Унификация поддержки в экосистеме ПО (компиляторы, симуляторы, библиотеки)
- Упрощение поддержки в СКЗИ

⇒ Необходимо выработать согласованный подход к custom RISC-V ГОСТ ISA в России

Потребность в стандартизации

РусКрипто

- Расширение номенклатуры RISC-V IP с поддержкой ГОСТ-расширения от разных вендоров
- Унификация поддержки в экосистеме ПО (компиляторы, симуляторы, библиотеки)
- Упрощение поддержки в СКЗИ

⇒ Необходимо выработать согласованный подход к custom RISC-V ГОСТ ISA в России

⇒ Выполнить необходимые условия для потенциальной конвертации в стандартное расширение RISC-V

Рабочая группа в Альянсе RISC-V

АЛЬЯНС RISC-V

РусКрипто

Цели:

- Обеспечение производительности ГОСТ-криптографии на архитектуре RISC-V наравне или выше зарубежных аналогов через расширения системы команд RISC-V
- Создание экосистемы для использования расширений

Текущий фокус:

- ГОСТ 31.12-2015 (блочные шифры Кузнечик, Магма)
- ГОСТ 31.11-2012 (криптографическая хэш-функция Стрибог)
- Скалярное и векторное расширения

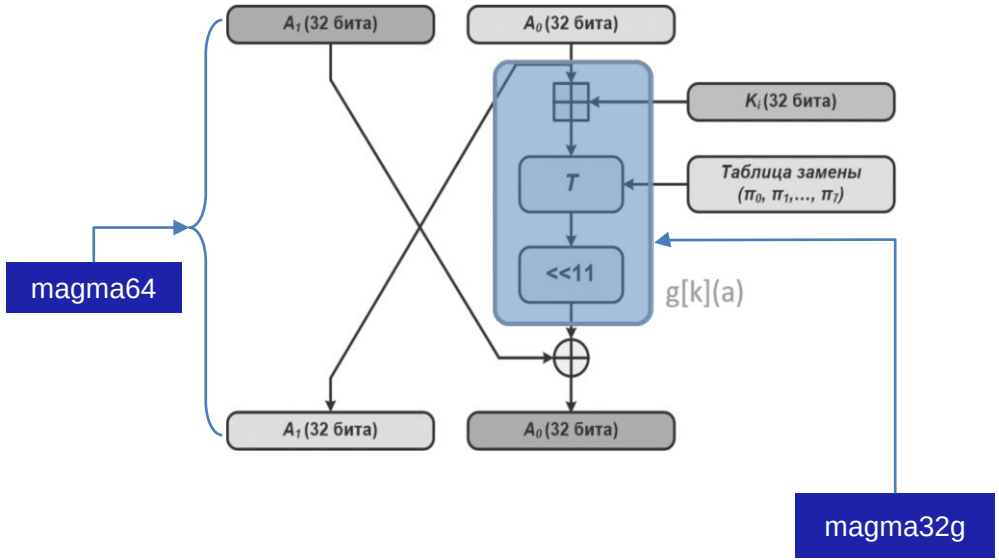
ТК Российского Альянса RISC-V,
РГ1 “Криптографические расширения RISC-V ISA”

Участников: 19
Компаний: 10

Xkgost (инструкции для Магмы)

РусКрипто

Схема раунда Магма



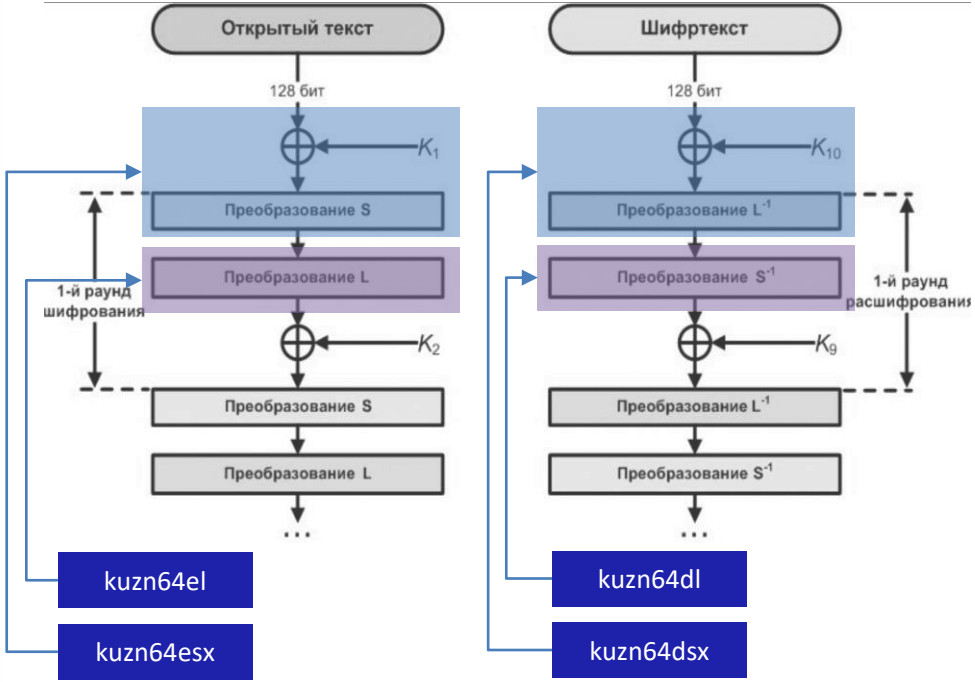
RV32
magma32g rd, rs1, rs2

RV64
magma64 rd, rs1, rs2

** Все инструкции обладают свойством постоянного времени исполнения*

Xkgost (инструкции для Кузнечика)

РусКрипто



RV32

kuzn32esx rd, rs1, rs2
kuzn32dsx rd, rs1, rs2
kuzn32el rd, rs1, rs2
kuzn32elh rd, rs1, rs2
kuzn32dl rd, rs1, rs2
kuzn32dlh rd, rs1, rs2

RV64

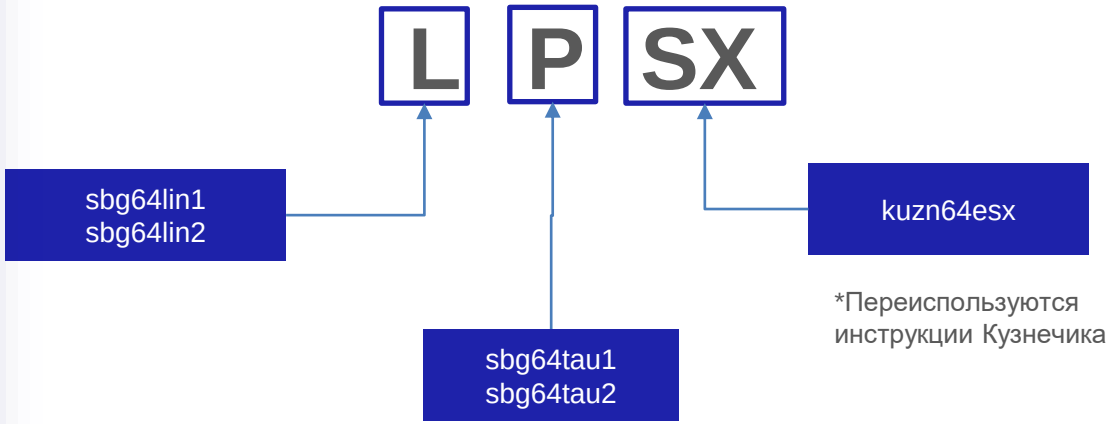
kuzn64esx rd, rs1, rs2
kuzn64dsx rd, rs1, rs2
kuzn64el rd, rs1, rs2
kuzn64dl rd, rs1, rs2

** Все инструкции обладают свойством постоянного времени исполнения*

Xkgost (инструкции для Стрибога)

РусКрипто

LPSX – основное преобразование алгоритма



*Переиспользуются инструкции Кузнечика

RV32

sbg32tau1 rd, rs1, rs2
sbg32tau2 rd, rs1, rs2
sbg32lin1 rd, rs1, rs2
sbg32lin2 rd, rs1, rs2

RV64

sbg64tau1 rd, rs1, rs2
sbg64tau2 rd, rs1, rs2
sbg64lin1 rd, rs1, rs2
sbg64lin2 rd, rs1, rs2

** Все инструкции обладают свойством постоянного времени исполнения*

Xkgost (экосистема)

РусКрипто

Спецификация расширения ISA:
- Семантика инструкций
- Мнемоники
- Кодировки (*custom*-пространство)

Поддержка в экосистеме
- Компиляторы
  - GCC / binutils
  - Clang
- Эмуляторы
  - Spike
  - QEMU
- Формальная модель описания расширения
  - SAIL
- Тесты

Поддержка в open-source крипто-библиотеках
- OpenSSL GOST Engine

2.17. sbg64tau1

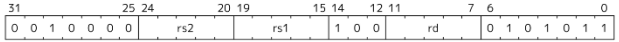
Synopsis

Interleave low-order bytes from two source registers.

Mnemonic

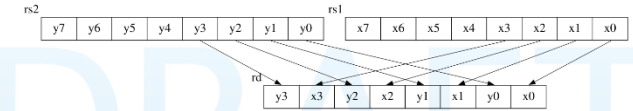
sbg64tau1 rd, rs1, rs2

Encoding



Description

This is an auxiliary instruction to perform a *P* transform (low-order bytes) of the hash function Streebog. Interleave the low-order bytes from `rs1` and `rs2` into the destination operand. The high-order bytes are ignored. This instruction must **always** be implemented that its execution latency does not depend on the data being operated on.



Operation

See Appendix C for the supporting Sail code.

```
function clause execute (SBG64TAU1(rs2, rs1, rd)) = {
  assert(sizeof(xlen) == 64);
  X(rd) = sbg_tau(X(rs1), X(rs2));
  RETIRE_SUCCESS
}
```

Included in

Extension	Minimum version	Lifecycle state
Xkgost (RV64)		

Пример использования

```c
#include <riscv_crypto_gost.h>

/* Encrypts one 64-bit block */
static inline void magma_encrypt_block(magma_ctx* ctx, const uint8_t* in, uint8_t* out)
{
    /* The magma{32g,64} instructions consume data
     * in little-endian format, no additional byte-swap is required
     */
    uint64_t state = *(uint64_t*)in;

    for (size_t j = 0; j < 3; ++j) {
        for (size_t i = 0; i < 8; ++i) {
            state = __riscv_magma64(state, ctx->key[i]);
        }
    }

    for (int i = 7; i >= 0; --i) {
        state = __riscv_magma64(state, ctx->key[i]);
    }

    uint64_t* output = (uint64_t*)out;
    *output          = swap_words(state);
}
```

```
clang -march=rv64gc_xkgost <…> magma.c
```

РусКрипто

Производительность Xkgost*

РусКрипто

	RV32	RV64	
	Тактов/байт	Тактов/байт	МБ/с @ 1ГГц
Магма	8.8	3.4	286
Кузнечик	14.5	1.6	584
Стрибог	53.3	10	95
AES-128	12.5	1.43	667

** Данные для ECB режимов блочных шифров*
RV32 – BM-310 (MDR1206FI) с аналогичным расширением Xgost
RV64 - BI-671 (ПЛИС) с аналогичным расширением Xgost

Что дальше?

- Ожидаем появление поддержки Xkgost в RISC-V IP российских вендоров

- Разработка векторного набора команд RISC-V для ГОСТ-криптографии в составе рабочей группы Альянса RISC-V

- Организация взаимодействия с ТК26 и Академией Криптографии

РусКрипто

Открытые вопросы

- Поддержка RISC-V и Xkgost в сертифицируемых СКЗИ

- Вопросы сертификации реализаций Xkgost для криптографии
  - Что необходимо сделать, что облегчить данный процесс

РусКрипто

РусКрипто

СПАСИБО
ЗА ВНИМАНИЕ