RISC-V и тестовые генераторы

МФТИ
Осень 2024

Кто преподает?

- Тарасов Юлий @botan_razdolb

- Белов Владислав @sick_hoof

Где преподает?

Аудитория – 3.28б (левая) Физтех.Цифра

Слайды – Github [riscv-technologies-lab/testgen-lectures]

Когда преподает?

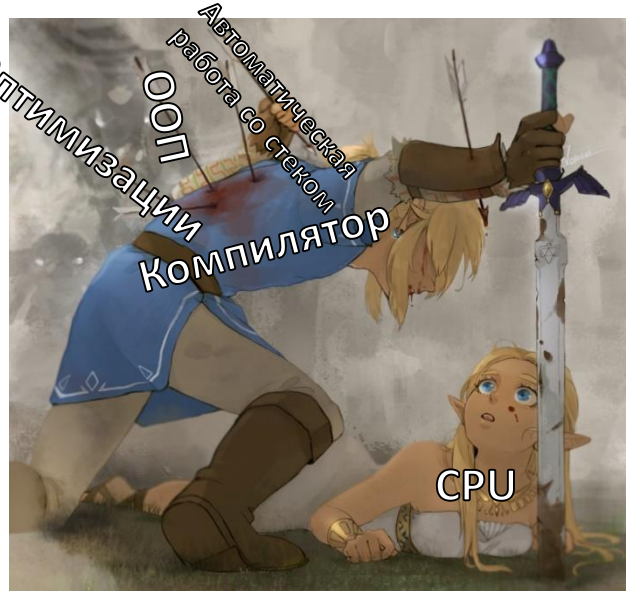
Каждый понедельник 10:45 – 12:10

Что преподает? (в этом семестре)

- Архитектура и экосистема RISC-V
- Как сделать свой RISC-V
- Как понять что ваш процессор работает (или нет)
- Почему важно готовиться к выпуску процессора и как это сделать
- Функциональный симулятор
- “Hello, world” на RISC-V
- Linux на RISC-V
- Как запустить DOOM на Linux, который запущен на вашем симуляторе

Уровни абстракции

- Абстракции помогают не сойти с ума
- SW (software) – компилятор и выше
- HW (hardware) – микроархитектура и ниже
- Архитектура – интерфейс между SW и HW



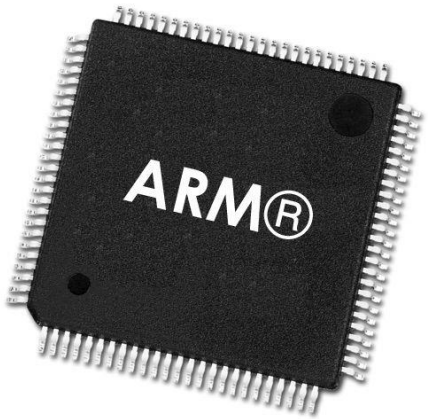
Архитектура определяет …

- Набор команд (как их семантику, так и кодировку)
- Регистры
- Типы данных
- Работу с памятью
- Интерфейсы, ввод/вывод
- Создание прерываний и исключительных состояний и их обработка

Какие архитектуры уже есть

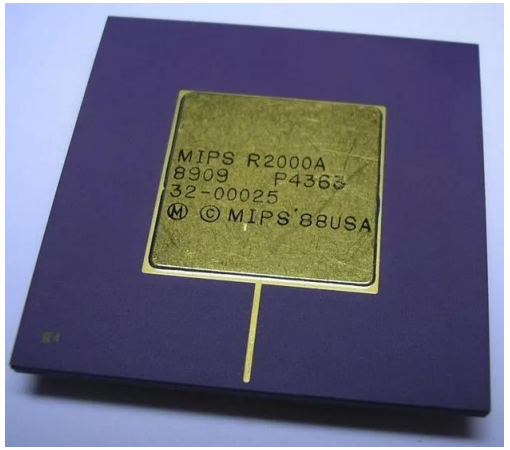
x86
x86-64

Ноутбуки, ПК, сервера



ARM

Мобильные устройства, сервера



MIPS

PSP, сетевые устройства

Эльбрус

ДАННЫЕ УДАЛЕНЫ

Зачем еще одна?

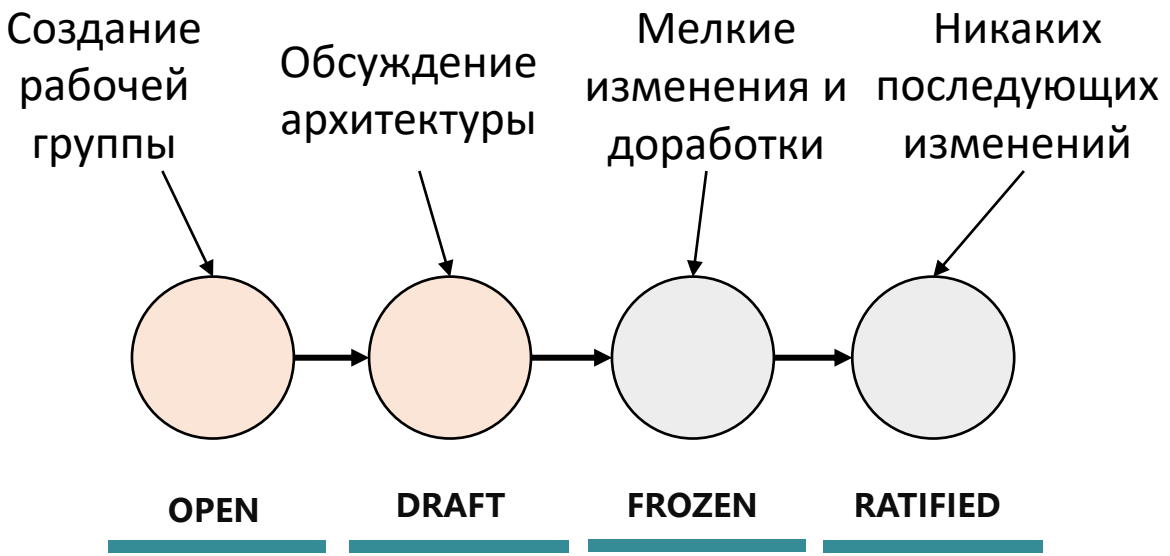
RISC-V

- Открытая
  - Спецификация в открытом доступе, royalty-free
- Свободная
  - Развитие в рамках комитета
- Модульная и расширяемая
  - Минимальный базовый набор
  - Дополнительная функциональность включается через расширения
  - Возможно включать расширения в различных комбинациях

RISC-V Foundation

- Основан в 2015 индустриальными лидерами и стартапами
- 3900+ членов из 70+ стран
- Продвигает исследования и инновации

Процесс ратификации спецификации RISC-V



Альянс RISC-V

Задачи:

- Создание открытого сообщества разработчиков
- Участие в фундаментальных исследованиях
- Развитие российской экосистемы продуктов

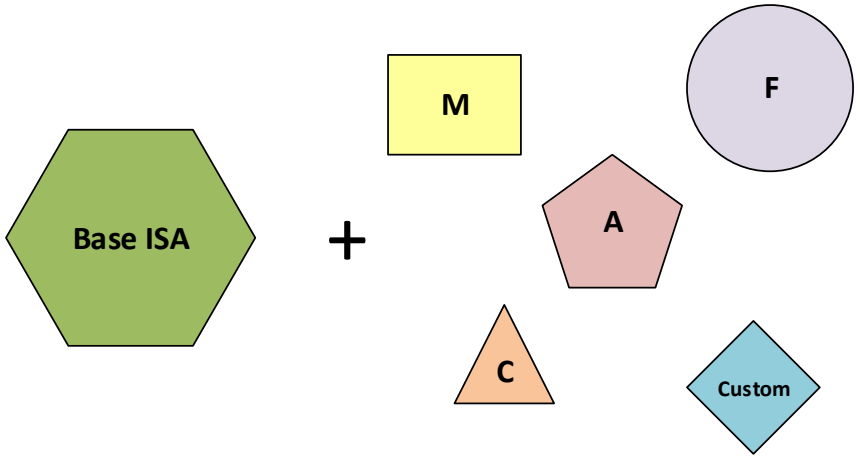
Профильные комитеты:

- Технологический
- Индустриальный
- Юридический
- Академический

Модульность и расширяемость RISC-V

- Базовый набор и стандартные расширения зафиксированы
- Добавление функциональности через расширения, не выпуск новых версий

M	Умножение и деление
C	Сжатые инструкции
F	Single-precision floats
D	Double-precision floats
E	Сокращенное количество регистров
A	Атомики
Z*	Другие стандартные расширения



Минимализм базовой ISA

Jumps & Calls
JAL
JALR
BEQ
BNE
BLT
BGE
BLTU
BGEU

Loads & Stores
LB
LH
LW
LBU
LHU
SB
SH
SW
LWU
LD
SD

Arithmetics			
ADD	ADDI	ADDW	ADDIW
SUB		SUBW	
OR	ORI		
XOR	XORI		
AND	ANDI		
SRL	SRLI	SRLW	SRLIW
SLL	SLLI	SLLW	SLLIW
SRA	SRAI	SRAW	SRAIW

Data flow			
SLT	SLTU	SLTI	SLTIU

Special
FENCE
ECALL
EBREAK

Upper immediate
LUI
AUIPC

Псевдооперации

```
for (i = 0; i < N; ++i)
  if (a[i] == x)
    return i;
```

```
        mv      a5,a0
        li      a0,0
        j       .loop
.latch:
        addiw   a0,a0,1
.loop:
        lw      a4,0(a5)
        addi    a5,a5,4
        bne     a4,a2,.latch
.exit:
        ret
```

Псевдоинструкция	Базовая инструкция	Смысл
`nop`	`addi x0, x0, 0`	Нет операции
`li rd, immediate`	*Различные последовательности*	Загрузка константы
`mv rd, rs`	`addi rd, rs, 0`	Копирование регистров
`not rd, rs`	`xori rd, rs, -1`	Инверсия числа
`neg rd, rs`	`sub rd, x0, rs`	Изменение знака числа
`seqz rd, rs`	`sltiu rd, rs, 1`	Установить 1, если == 0
`snez rd, rs`	`sltu rd, x0, rs`	Установить 1, если != 0
`sltz rd, rs`	`slt rd, rs, x0`	Установить 1, если < 0
`sgtz rd, rs`	`slt rd, x0, rs`	Установить 1, если > 0
`beqz rs, offset`	`beq rs, x0, offset`	Перейти, если == 0
`bnez rs, offset`	`bne rs, x0, offset`	Перейти, если != 0
`blez rs, offset`	`bge x0, rs, offset`	Перейти, если <= 0
`bgez rs, offset`	`bge rs, x0, offset`	Перейти, если >= 0
`bltz rs, offset`	`blt rs, x0, offset`	Перейти, если < 0
`bgtz rs, offset`	`blt x0, rs, offset`	Перейти, если > 0
`bgt rs1, rs2, offset`	`blt rs2, rs1, offset`	Перейти, если >
`ble rs1, rs2, offset`	`bge rs2, rs1, offset`	Перейти, если <=
`bgtu rs1, rs2, offset`	`bltu rs2, rs1, offset`	Перейти, если >, беззнаковое
`bleu rs1, rs2, offset`	`bgeu rs2, rs1, offset`	Перейти, если <=, беззнаковое
`j offset`	`jal x0, offset`	Переход по метке
`jal offset`	`jal x1, offset`	Переход с сохранением адреса возврата
`jr rs`	`jalr x0, 0(rs)`	Переход по значению из регистра
`jalr rs`	`jalr x1, 0(rs)`	Переход с сохранением адреса возврата
`ret`	`jalr x0, x1, 0`	Возврат из подпрограммы

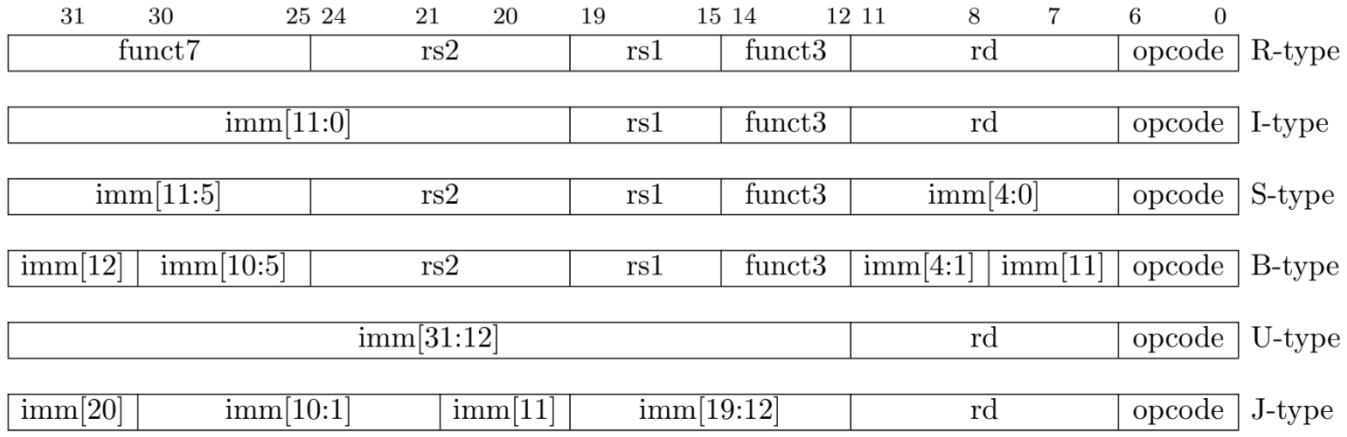
Пример кодировки инструкции

imm[12]	imm[10:5]						rs2					rs1					funct3			imm[4:1]				imm[11]	opcode						
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	1	1	1	0	0	0	0	1	0	1	0	0	0	0	0	0	1	0	1	0	1	0	0	1	1	1	0	0	0	1	1

Декодер в твоей голове:

- Первые 6 бит – уникальный идентификатор типа инструкции (опкод)
- Каждая инструкция принадлежит какому-то типу
- Знание о типе инструкции даёт всю необходимую информацию для декода инструкции

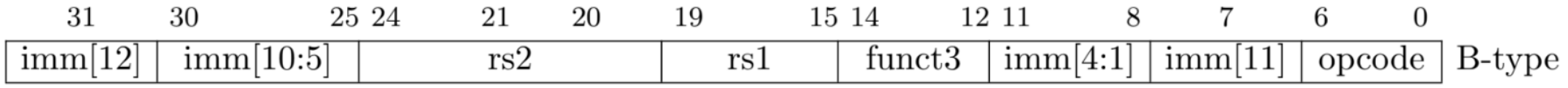
Формат кодировки



Пример кодировки инструкции

imm [12]	imm[10:5]						rs2					rs1					funct3			imm[4:1]				imm [11]	opcode						
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	1	1	1	0	0	0	0	1	0	1	0	0	0	0	0	0	1	0	1	0	1	0	0	1	1	1	0	0	0	1	1

Branch

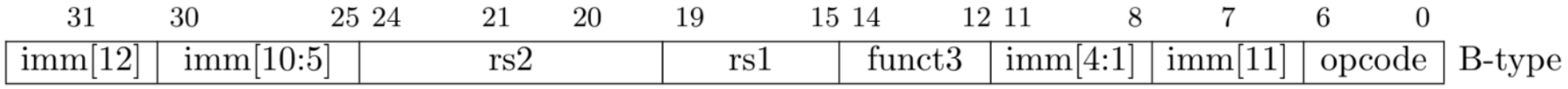


Пример кодировки инструкции

imm [12]	imm[10:5]						rs2					rs1					funct3			imm[4:1]				imm [11]	opcode						
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	1	1	1	0	0	0	0	1	0	1	0	0	0	0	0	0	1	0	1	0	1	0	0	1	1	1	0	0	0	1	1

BGE

```
bge ?, ?, ?
```



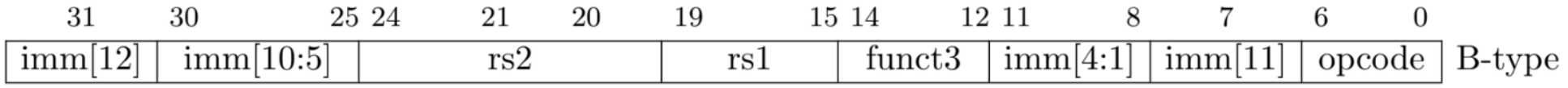
Пример кодировки инструкции

imm [12]	imm[10:5]						rs2					rs1					funct3			imm[4:1]				imm [11]	opcode						
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	1	1	1	0	0	0	0	1	0	1	0	0	0	0	0	0	1	0	1	0	1	0	0	1	1	1	0	0	0	1	1

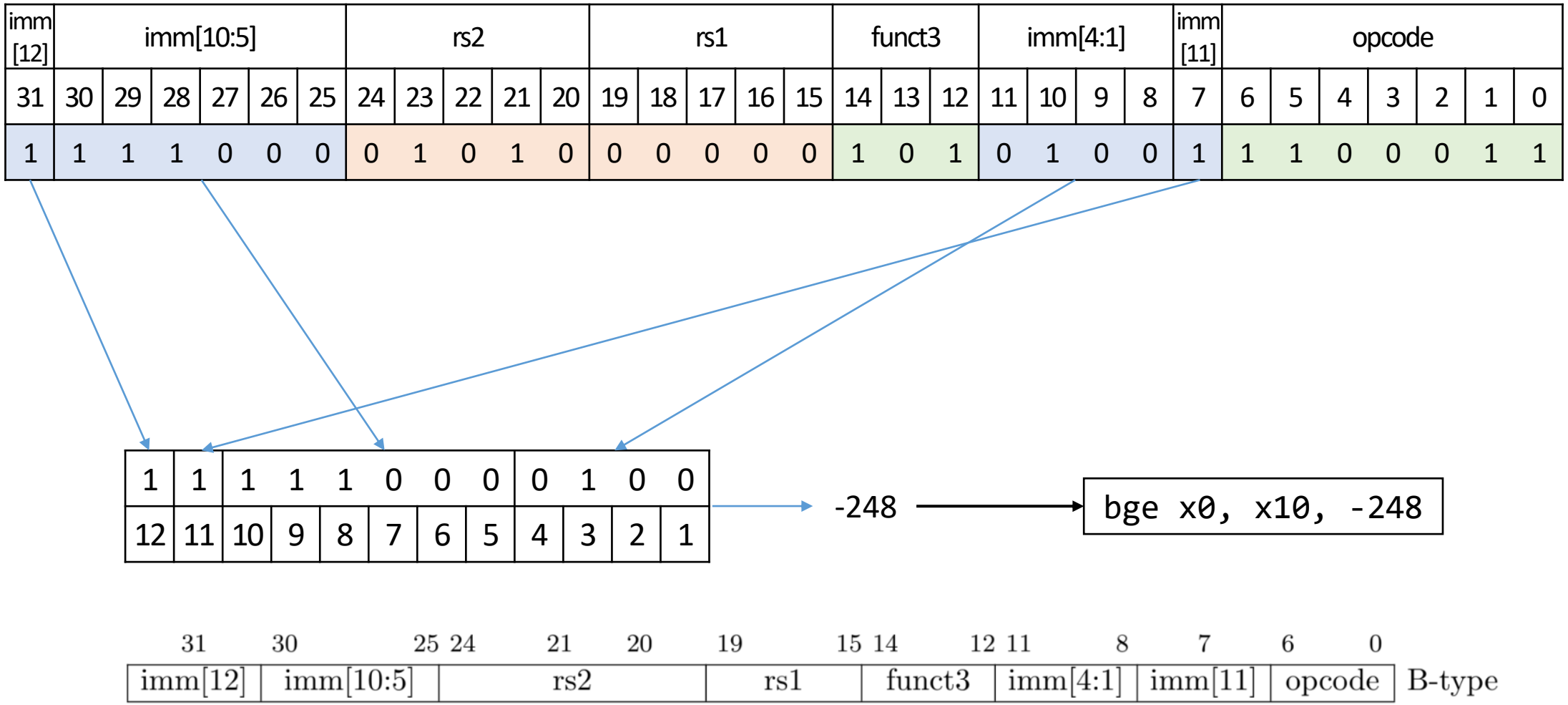
rs2 = x10

rs1 = x0

```
bge x0, x10, ?
```



Пример кодировки инструкции



Пример кодировки инструкции

imm [12]	imm[10:5]						rs2					rs1					funct3			imm[4:1]				imm [11]	opcode						
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	1	1	1	0	0	0	0	1	0	1	0	0	0	0	0	0	1	0	1	0	1	0	0	1	1	1	0	0	0	1	1

```
bge x0, x10, -248
```

Обсуждение:

- Знаете ли вы ещё какие-нибудь кодировки?
- Подумайте о плюсах и минусов такого формата кодировки
- Как бы вы написали программу, которая декодировала бы инструкции RISC-V ISA?

ABI и calling convention

Name	Alias
x0	zero
x1	ra
x2	sp
x3	gp
x4	tp
x5-x7	t0-t2
x8-x9	s0-s1
x10-x15,x16,x17	a0-a5, a6,a7
x18-x27	s2-s11
x28-x31	t3-t6

FP Name	FP Alias
f0-f7	ft0-ft7
f8-f9	fs0-fs1
f10-f17	fa0-fa7
f18-f27	fs2-fs11
f28-f31	ft8-ft11

Красных регистров нет в RV32E

Обсуждение

Оказавшись компилятором, чтобы вы сказали?

- Для RV32I?
- Для RV32IF?
- Для RV32ID?

```
int fto_int(float f) {
  return f;
}
```

Гладко было на бумаге…

Не компилируется, потому что нет fp регистров … но это можно заставить скомпилироваться: godbolt.org

Доступные RISC-V ABI: **ilp32 ilp32d ilp32e ilp32f lp64 lp64d lp64f**

LP64 – long и pointer = 64, int = 32

LP64d – то же и поддержаны double

ILP32, **ILP32d** – integer, long и pointer = 32

А с каким ABI собрана libc, с которой вы линкуетесь? :-)

А с какими расширениями собрана libc, с которой вы линкуетесь?

To be continued …

На следующем занятии прикоснемся к экосистеме RISC-V:

- Научимся пользоваться кросс-компилятором (и узнаем почему он кросс)
- Научимся запускать RISC-V программы без RISC-V
- Запустим программу на настоящем RISC-V

Подготовка к следующему занятию

- Toolchain для кросс-платформенной разработки состоит из большого числа инструментов
- Их совместная настройка может требовать заметных усилий
- Для решения подобной проблемы была создана технологии контейнеризации приложений
- На наших занятиях мы рекомендуем использовать docker с настроенными инструментами и окружением

Пару слов про docker

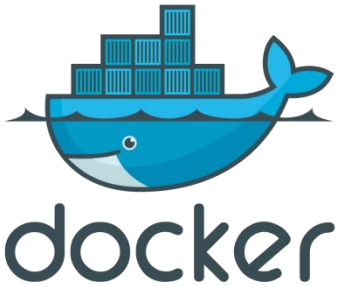
Docker – система упаковывания приложения вместе с окружением в «контейнер», а также управления такими контейнерами.

GameBoy:

- Просто вставь картридж и играй
- Хочешь поделиться с другом игрой – просто передай картридж

Docker позволяет передавать контейнер, в котором приложение будет сразу работать

Пару слов про docker



Окружение приложения:

- Бинарные зависимости (библиотеки, другие приложения)
- Структура файловой системы
- Конфигурация системы
- Переменные окружения

Задание к следующему занятию

- Настроить docker и скачать образ контейнера с Syntacore Development Toolkit (SC-DT) по инструкциям:
  - Docker инструкция
  - SC-DT инструкция

Или

- Самостоятельно нативно установить SC-DT с официального сайта

Список литературы

- The RISC-V Instruction Set Manual Volume I Unprivileged Architecture Version 20240411 // Chapter 2. RV32I Base Integer Instruction Set, Version 2.1
- Лекция на неофициальном открытии лаборатории RISC-V технологий в МФТИ, февраль 2024: https://youtu.be/xY_Ne9ZznJ4
- Hennessy J. L., Patterson D. A. Computer architecture: a quantitative approach. – Morgan kaufmann, 2017.
- Инструменты программирования для открытой архитектуры RISC V. Константин Владимиров, конференция True Tech, 2024: https://youtu.be/qoNjayusCX4
- Расширяемая архитектура RISC-V и Syntacore SW Tools. Константин Владимиров, конференция «Салют, OS DevConf!», 2024: https://youtu.be/1zLxxxLc0xI

Канал курса в telegram