

Средства симуляции ЦП и ОС и изучение поведения программ

Лекция №3



Державин Андрей
Шурыгин Антон

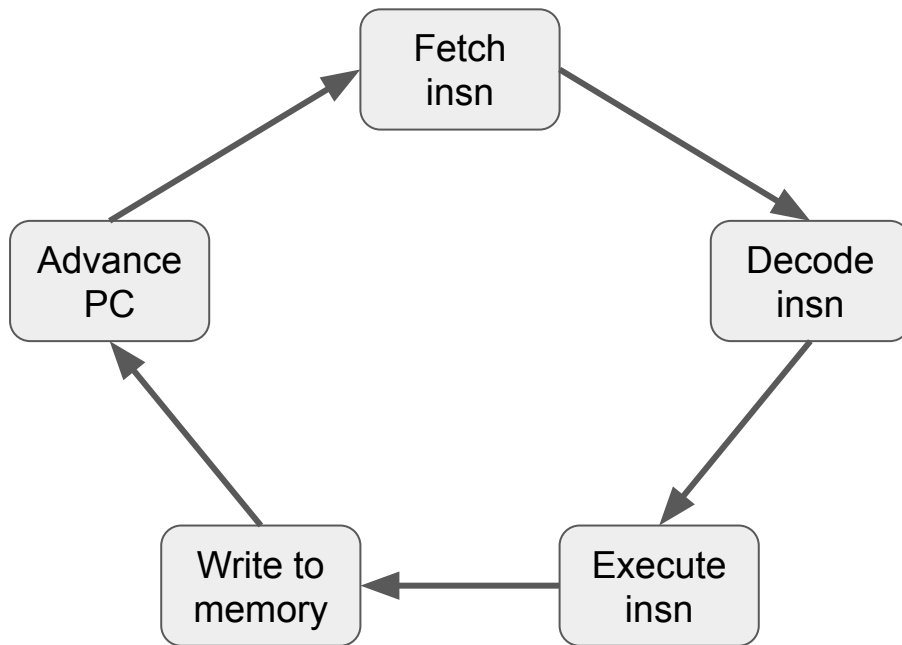
➤ **Квиз**

- Процедура декодирования
- Архитектура RISC-V
- Дерево декодирования
- Домашнее задание №2

- Квиз
- **Процедура декодирования**
 - Архитектура RISC-V
 - Дерево декодирования
 - Домашнее задание №2

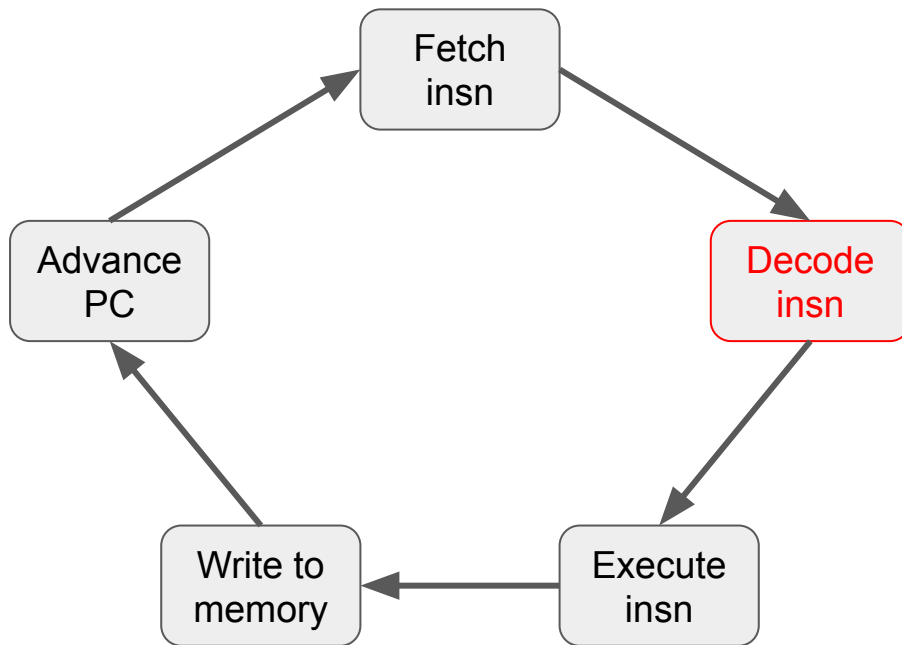
Стадии интерпретатора

- 5 стадий интерпретации



Стадии интерпретатора

- 5 стадий интерпретации



Декодирование

```
Instruction decode(Register bytes) {  
    Instruction insn{.opcode=get_opcode(bytes)};  
    switch (insn.opc) {  
        case Opcode::kAdd:  
            insn.src1 = get_src1(bytes);  
            insn.src2 = get_src2(bytes);  
            insn.dst = get_dst(bytes);  
            break;  
    }  
    return insn;  
}
```

Анализ выражений

letter = [a-zA-Z]
digit = [0-9]
digits = digit digit*

identifier = letter | (letter | digit | _)*

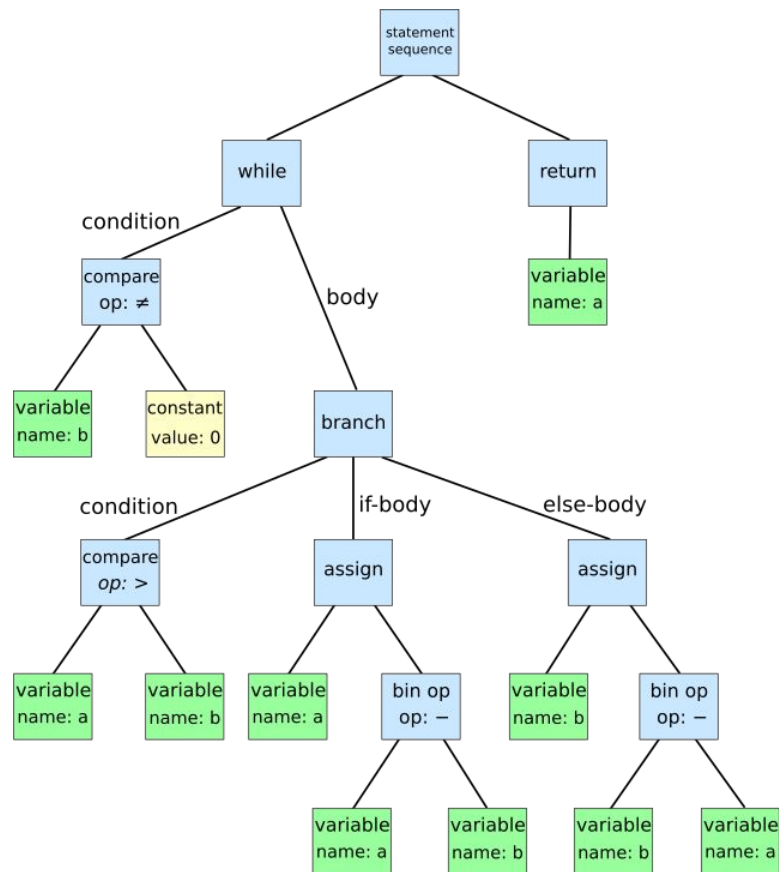
fraction = .digits | ϵ

exponent = ((E | e) (+ | - | ϵ)) digits) | ϵ

number = digits fraction exponent

operator = + | - | * | / | > | >= | < | <= | = | ==

parenthesis = (|)



Декодирование как анализ

- Машинное представление инструкций – тоже всего лишь язык
- Особенности языка, позволяющие строить декодеры оптимально:



Декодирование как анализ

- Машинное представление инструкций – тоже всего лишь язык
- Особенности языка, позволяющие строить декодеры оптимально:
 - Длина инструкций
 - Префиксный код
 - Режим процессора



Процедура декодирования

- В реальной процессоре декодер – это блок из логических элементов
- В функциональном симуляторе – процедура на языке программирования

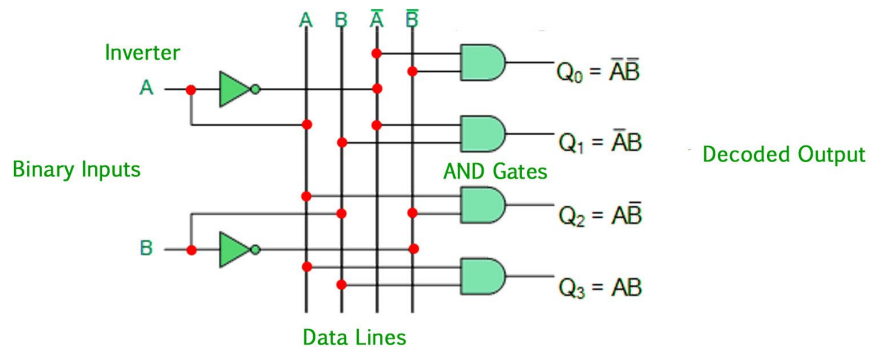
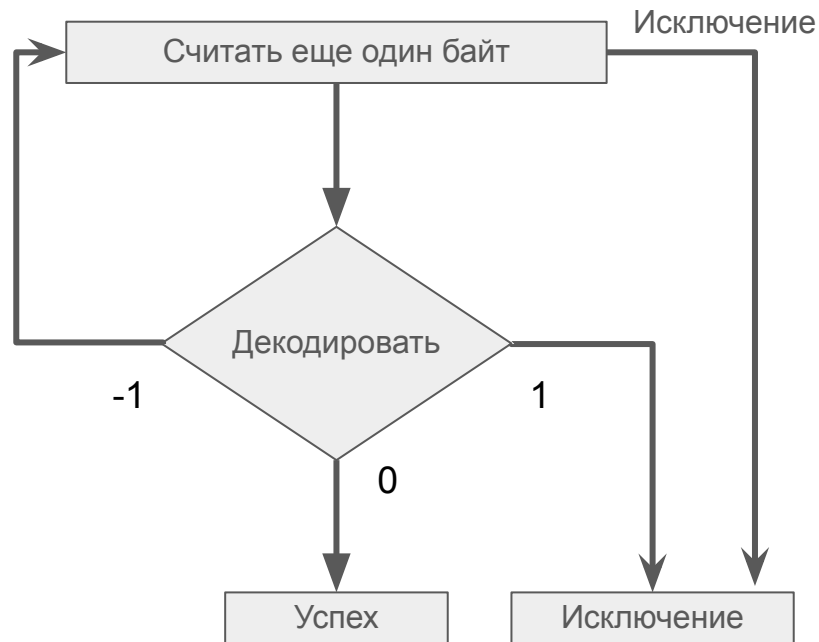


Схема двоичного декодера 2-4

Процедура декодирования

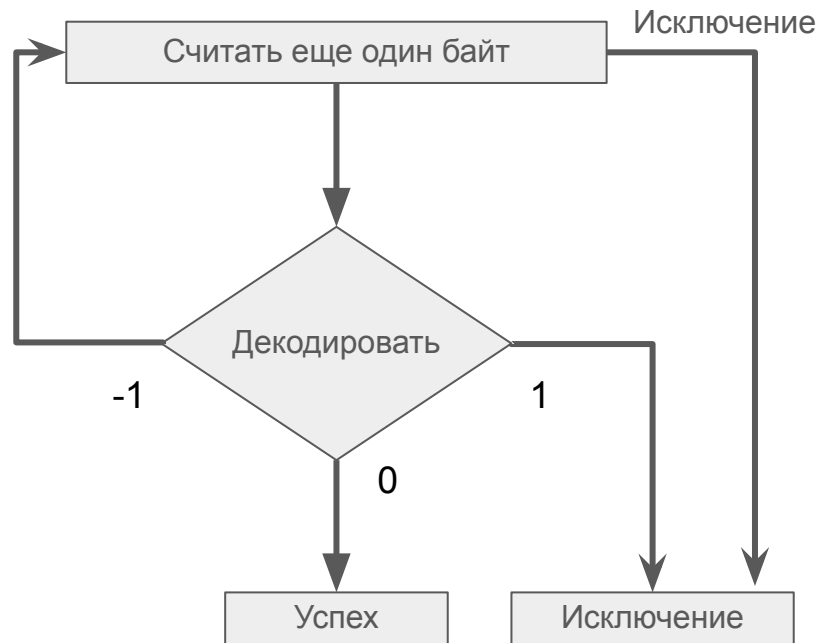
- Вход – массив байт, полученный на фазе Fetch
- Выход:
 - Код возврата декодирования
 - Поля результата



Блок-схема процедуры декодирования для машинного слова переменной длины

Процедура декодирования

- Вход – массив байт, полученный на фазе Fetch
- Выход:
 - ✓ Код возврата декодирования
 - Поля результата
 - Код операции
 - Информация об операндах
 - Длина инструкции
 - Дополнительная информация



Блок-схема процедуры декодирования для машинного слова переменной длины

Декодирование

```
Instruction decode(Register bytes) {  
    Instruction insn{.opcode=get_opcode(bytes)};  
    switch (insn.opc) {  
        case Opcode::kAdd:  
            insn.src1 = get_src1(bytes);  
            insn.src2 = get_src2(bytes);  
            insn.dst = get_dst(bytes);  
            break;  
    }  
    return insn;  
}
```

Декодирование

```
Instruction decode(Register bytes) {  
    Instruction insn{.opcode=get_opcode(bytes)};  
    switch (insn.opc) {  
        case Opcode::kAdd:  
            insn.src1 = get_src1(bytes);  
            insn.src2 = get_src2(bytes);  
            insn.dst = get_dst(bytes);  
            break;  
    }  
    return insn;  
}
```

- Квиз
- Процедура декодирования

➤ **Архитектура RISC-V**

- Дерево декодирования
- Домашнее задание №2

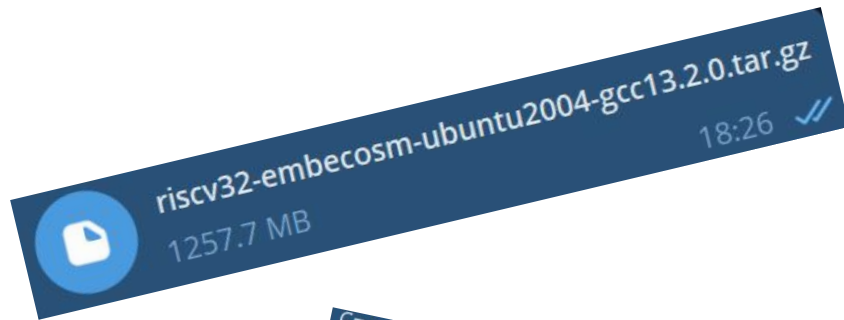
Архитектура RISC-V

- Что такое RISC-V?
 - Открытая, свободная и расширяемая система



Архитектура RISC-V

- Что такое RISC-V?
- Toolchain RISC-V и симулятор QEMU



Спасибо за найденную проблему, мы немного накосячили с тулчейном.



RV32I

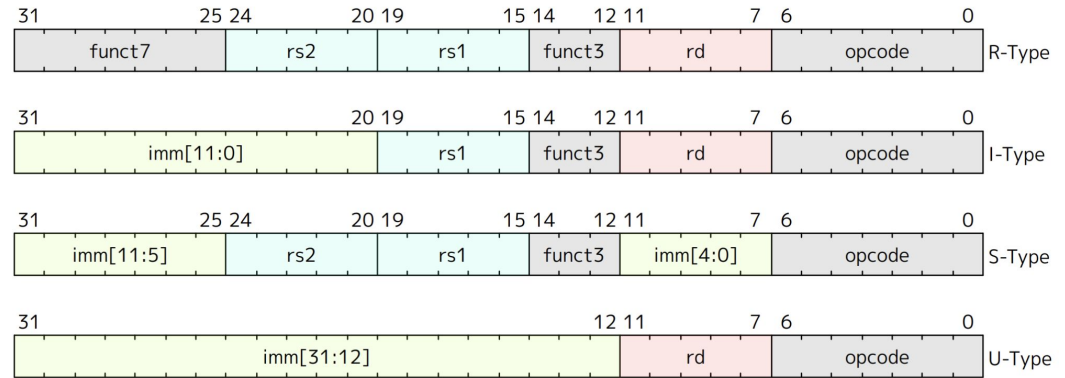
- Регистры

- Ширина каждого регистра – 32 бита
- Регистр x0 – аппаратный ноль
- Регистры общего назначения x1 - x31
- PC (англ. program counter) – регистр, хранящий адрес текущей инструкции



Система команд RV32I

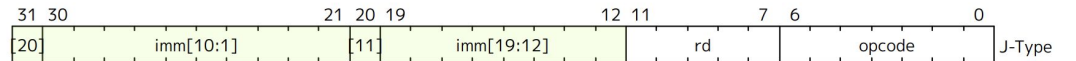
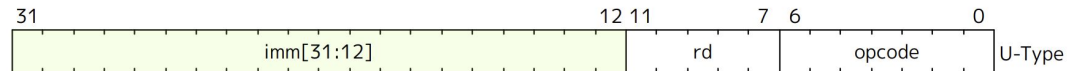
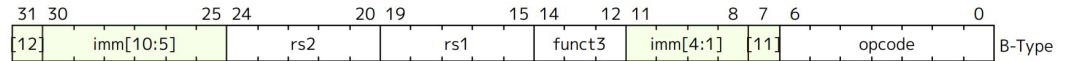
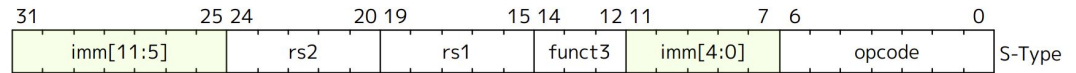
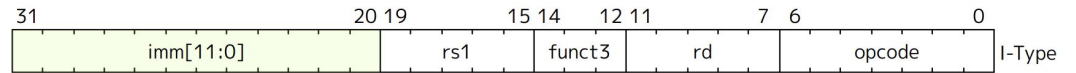
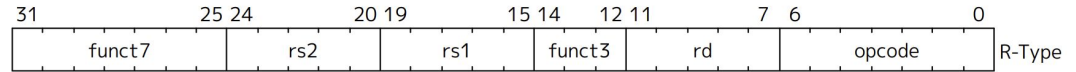
- В RV32I четыре основных формата инструкций:
 - R – register type
 - I – immediate type
 - S – store type
 - U – type
- Фиксированная длина инструкции – 32 бита
- Выравнивание в памяти и IALIGN=32



Основные типы кодировок RV32I

Система команд RV32I

- Особенности работы с immediate
 - B – branch type
 - J – jump type
- Различие B-type от S-type и U-type от J-type
- Знаковое расширение



Теперь точно основные типы кодеровок RV32I

Система команд RV32I

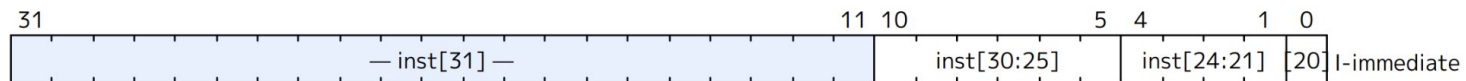
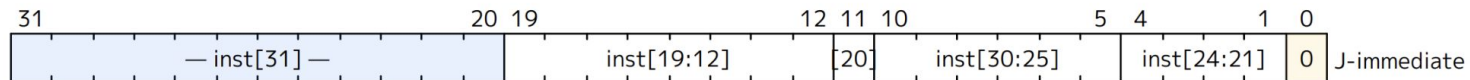
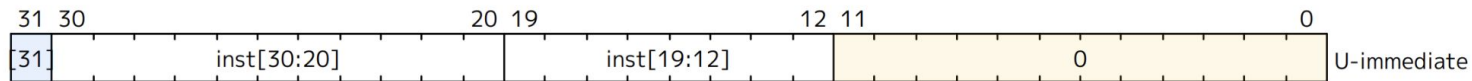
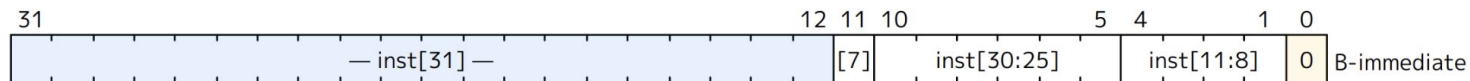
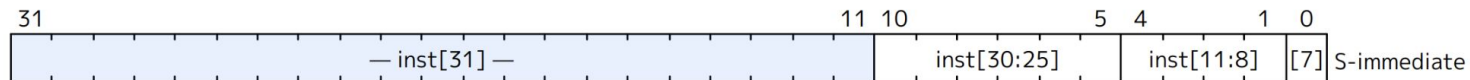


Figure 1. Types of immediate produced by RISC-V instructions.



Система команд RV32I

CONTROL FLOW

JAL
JALR

BNE
BLT
BEQ
BGE
BLTU
BGEU

STORE/LOAD

LB
LH
LW
LBU
LHU
SB
SH
SW

ARITHMETIC

ADD ADDI
SUB SUBI
OR ORI
XOR XORI
AND ANDI
SRL SRLI
SLL SLLI
SRA SRAI

DATA FLOW

SLT
SLTI
SLTIU

SYSTEM

ECALL
EBREAK
FENCE

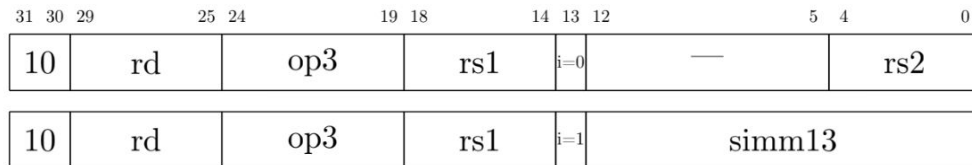
UPPER IMM

LUI
AUIPC

- Квиз
- Процедура декодирования
- Архитектура RISC-V
- **Дерево декодирования**
- Домашнее задание №2

Распознавание шаблонов

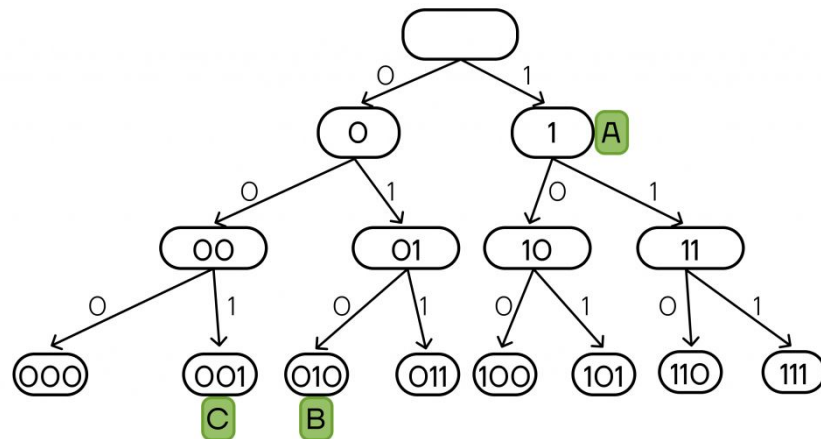
- Сопоставление массива байт набору шаблонов из документации архитектуры
- Любой входной последовательности бит соответствует минимум один шаблон



Шаблон кодировки архитектуры SPARC

Оптимизация декодирования

- Число шаблонов для сложной архитектуры может быть большим
- Переход от линейного декодера к дереву поиска



- Квиз
- Процедура декодирования
- Архитектура RISC-V
- Дерево декодирования
- **Домашнее задание №2**

Домашнее задание №2

- Реализуйте интерпретатор RISCV32I. Команды могут поступать на вход из массива (не elf-файл).
- В качестве инструкции завершения можно использовать EBREAK
- ECALL, FENCE не обязательны к реализации
- Интерпретировать байт-код программы поиска n-ого числа Фибоначчи

Домашнее задание №2

CONTROL FLOW

JAL
JALR

BNE
~~BLT~~
BEQ
~~BGE~~
~~BLTU~~
~~BGEU~~

STORE/LOAD

LB
LH
LW
~~LBU~~
~~LHU~~
SB
SH
SW

ARITHMETIC

ADD ADDI
SUB SUBI
~~OR ORI~~
~~XOR XORI~~
~~AND ANDI~~
~~SRL SRLI~~
~~SLL SLLI~~
~~SRA SRAI~~

DATA FLOW

~~SLT~~
~~SLTI~~
~~SLTIU~~

SYSTEM

~~ECALL~~
EBREAK
~~FENCE~~

UPPER IMM

LUI
AUIPC

Литература

1. [Simulation Foundations Book, in Russian, Grigory Rechistov](#)
2. [The RISC-V Instruction Set, Manual Volume I, Unprivileged Architecture](#)