Средства симуляции ЦП и ОС и изучение поведения программ

Вводная лекция

Державин Андрей
Шурыгин Антон

Преподаватели

Державин Андрей, 6 курс ФРКТ

- Функциональная симуляция
- Микроархитектура CPU

Шурыгин Антон, 6 курс ФРКТ

- Бинарная инструментация
- Дизайн архитектуры, симуляция

План

- Стек компьютерных технологий
- Компиляторы
- Архитектура набора команд
- Симуляторы и бинарные трансляторы

Уровни абстракций

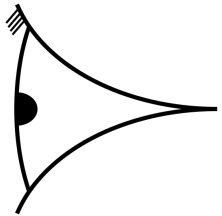
программный уровень	Приложения	Браузер, компьютерные игры
	Алгоритмы	Сортировка, поиск
	Язык программирования	C/C++, Java, Rust, Go
	Операционная система	Windows, Linux, Android
	Набор команд	x86, ARM, RISC-V
аппаратный уровень	Микроархитектура	Кэши, предсказание переходов
	Интегральные схемы	RTL,низкоуровневое описание
	Цифровая логика	Транзисторы, логические элементы
	Физический уровень	Полупроводники, электроны

Уровни абстракций

программный уровень

аппаратный уровень

- Приложения
- Алгоритмы
- Язык программирования
- Операционная система
- Набор команд
- Микроархитектура
- Интегральные схемы
- Цифровая логика
- Физический уровень



Золотое правило computer science

- Производительность

$$Perf = \frac{1}{Time} = \frac{1}{N_{instr} \cdot CPI \cdot T_{cycle}} = \frac{1}{N_{instr}} \cdot IPC \cdot f$$

- Время исполнения

$$Time = T_{cycle} \cdot Cycles\ Num = N_{instr} \cdot CPI \cdot T_{cycle}$$

- Как улучшить производительность?

$$Perf = \frac{1}{N_{instr}} \cdot IPC \cdot f$$

1. **IPC** - среднее количество инструкций, выполняемых за каждый такт
2. **CPI** - величина, обратная IPC.
3. **f** - тактовая частота

Золотое правило computer science

- Производительность

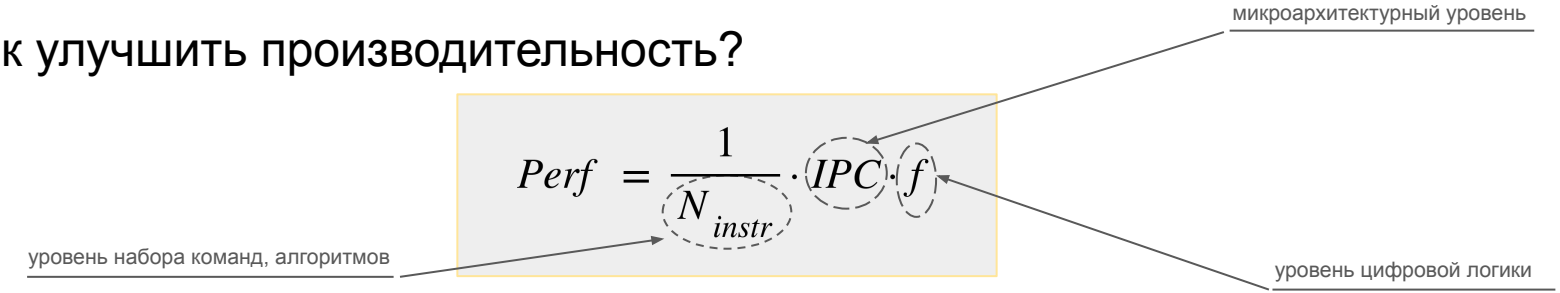
$$Perf = \frac{1}{Time} = \frac{1}{N_{instr} \cdot CPI \cdot T_{cycle}} = \frac{1}{N_{instr}} \cdot IPC \cdot f$$

- Время исполнения

$$Time = T_{cycle} \cdot Cycles\ Num = N_{instr} \cdot CPI \cdot T_{cycle}$$

1. **IPC** - среднее количество инструкций, выполняемых за каждый такт
2. **CPI** - величина, обратная IPC.
3. **f** - тактовая частота

- Как улучшить производительность?



Применение «золотой формулы»

1. **IPC** - среднее количество инструкций, выполняемых за каждый такт
2. **IC** - количество инструкций.
3. **f** - тактовая частота.

программный уровень

- Приложения
- Алгоритмы
- Язык программирования
- Операционная система

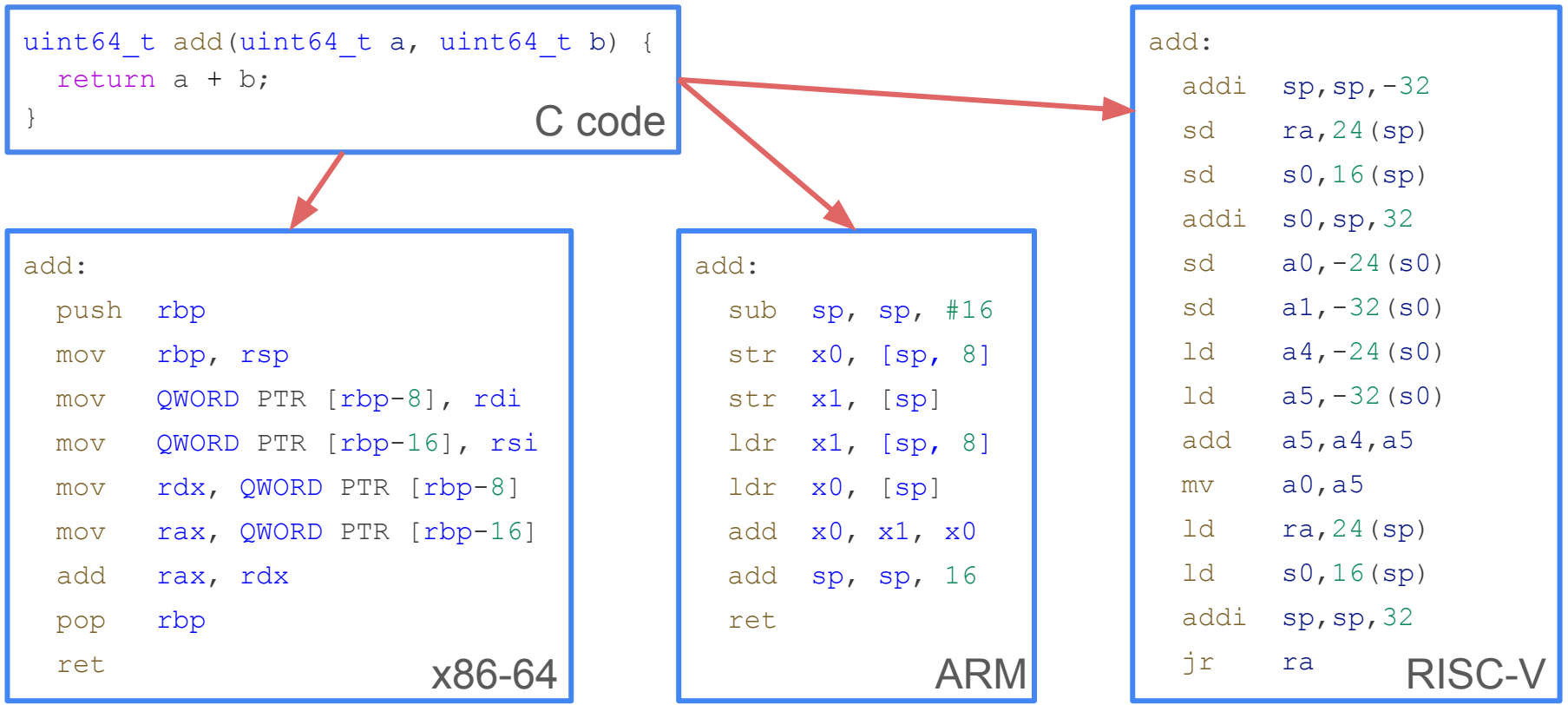
- Набор команд

аппаратный уровень

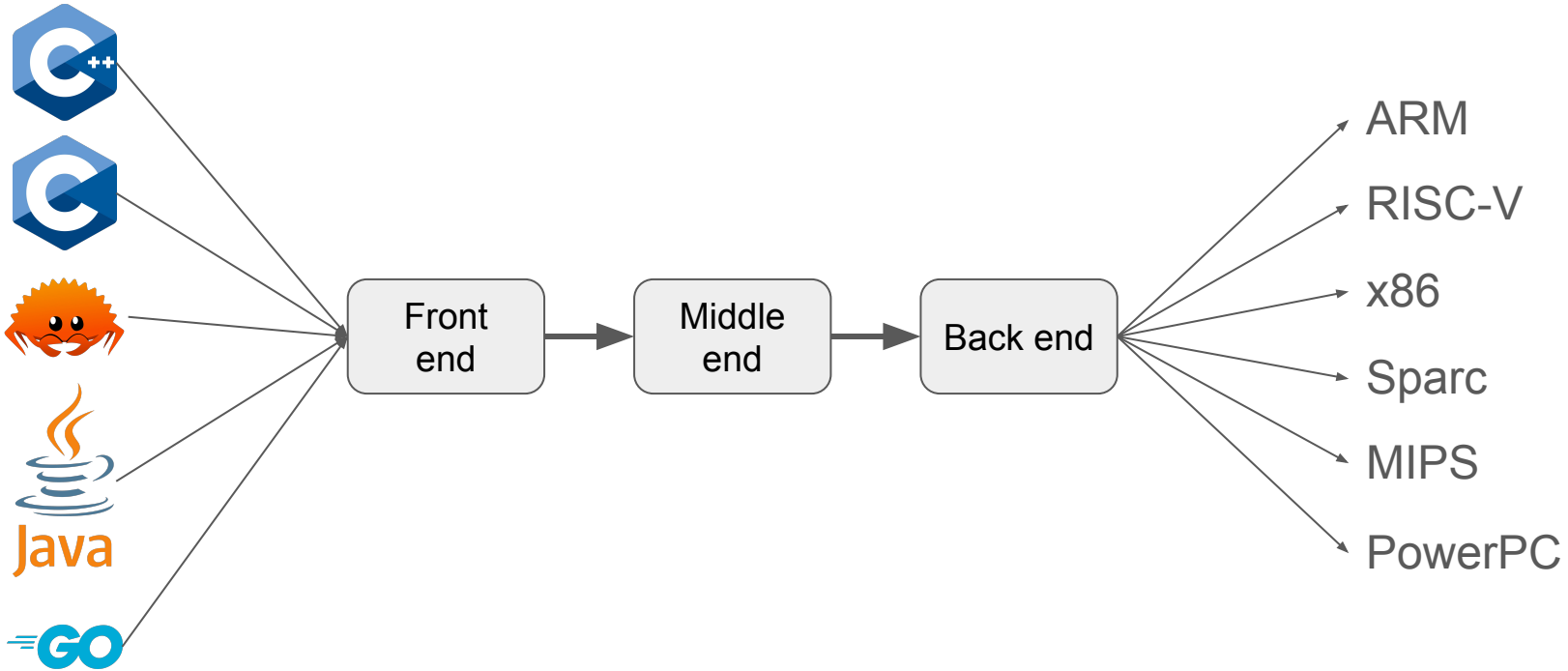
- Микроархитектура
- Интегральные схемы
- Цифровая логика
- Физический уровень

N_{instr}	IPC	f
+	-	-
+	±	-
+	±	-
+	±	-
+	+	-
-	+	+
-	±	+
-	±	+
-	±	+

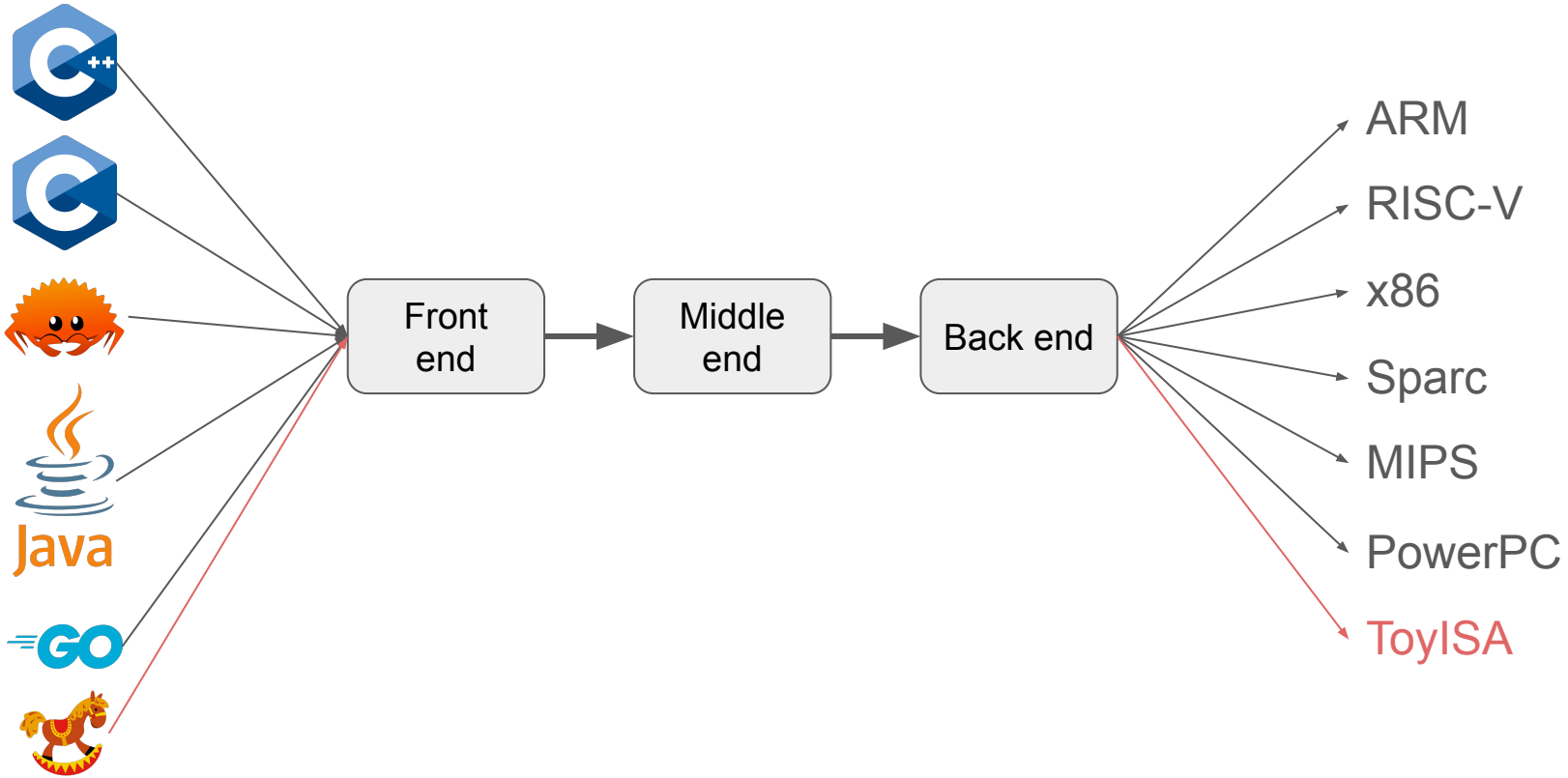
ISA


uint64_t add(uint64_t a, uint64_t b) {
return a + b;
}
C code
add:
push rbp
mov rbp, rsp
mov QWORD PTR [rbp-8], rdi
mov QWORD PTR [rbp-16], rsi
mov rdx, QWORD PTR [rbp-8]
mov rax, QWORD PTR [rbp-16]
add rax, rdx
pop rbp
ret
x86-64
add:
sub sp, sp, #16
str x0, [sp, 8]
str x1, [sp]
ldr x1, [sp, 8]
ldr x0, [sp]
add x0, x1, x0
add sp, sp, 16
ret
ARM
add:
addi sp,sp,-32
sd ra,24(sp)
sd s0,16(sp)
addi s0,sp,32
sd a0,-24(s0)
sd a1,-32(s0)
ld a4,-24(s0)
ld a5,-32(s0)
add a5,a4,a5
mv a0,a5
ld ra,24(sp)
ld s0,16(sp)
addi sp,sp,32
jr ra
RISC-V

Архитектура компилятора

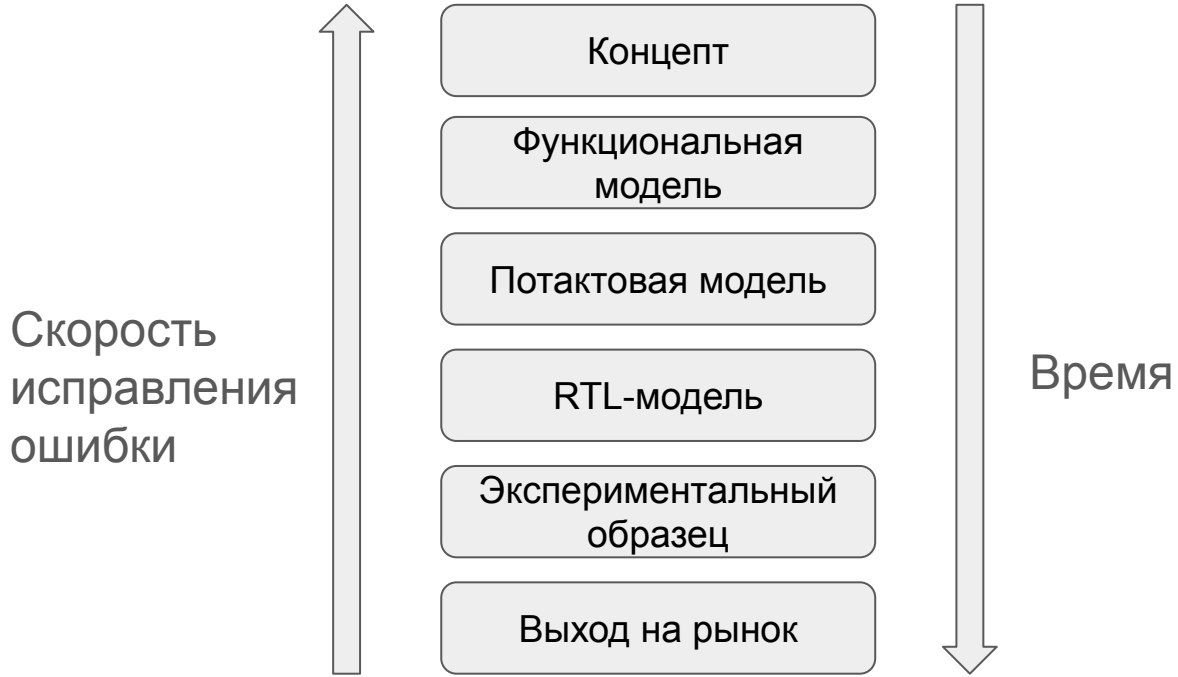


Архитектура компилятора



Моделирование систем

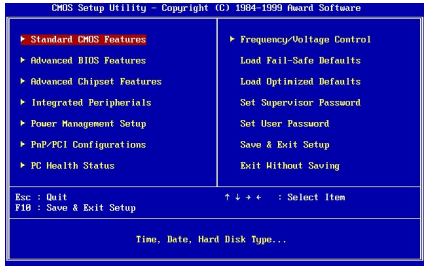
- Разработка вычислительных систем – сложный и длительный процесс:
  - Архитектура, дизайн, обратная совместимость, выявление ошибок на ранних этапах



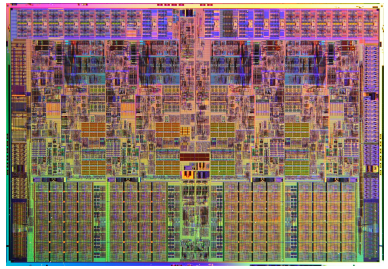
Моделирование систем

- Разработка моделей как способ понять поведение устройства
- Области применения моделей компьютера

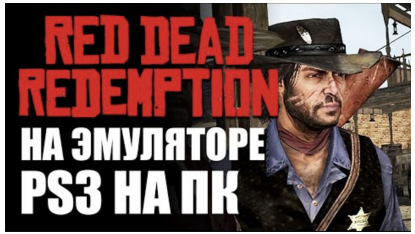
Обнаружение ошибок проектирования



Разработка ПО под аппаратуру



Исследование пространства проектирования



Выполнение программ не «неродной» архитектуре

Моделирование систем

- Разработка вычислительных систем – сложный и длительный процесс:
  - Архитектура, дизайн, обратная совместимость, выявление ошибок на ранних этапах

Понятие транслятора

вход/выход	Бинарный файл	Исполнение
Текстовый файл		
Бинарный файл		

Понятие транслятора

вход/выход	Бинарный файл	Исполнение
Текстовый файл	Компилятор	
Бинарный файл		

Понятие транслятора

вход/выход	Бинарный файл	Исполнение
Текстовый файл	Компилятор	Интерпретатор
Бинарный файл		

Понятие транслятора

вход/выход	Бинарный файл	Исполнение
Текстовый файл	Компилятор	Интерпретатор
Бинарный файл	Бинарный транслятор	

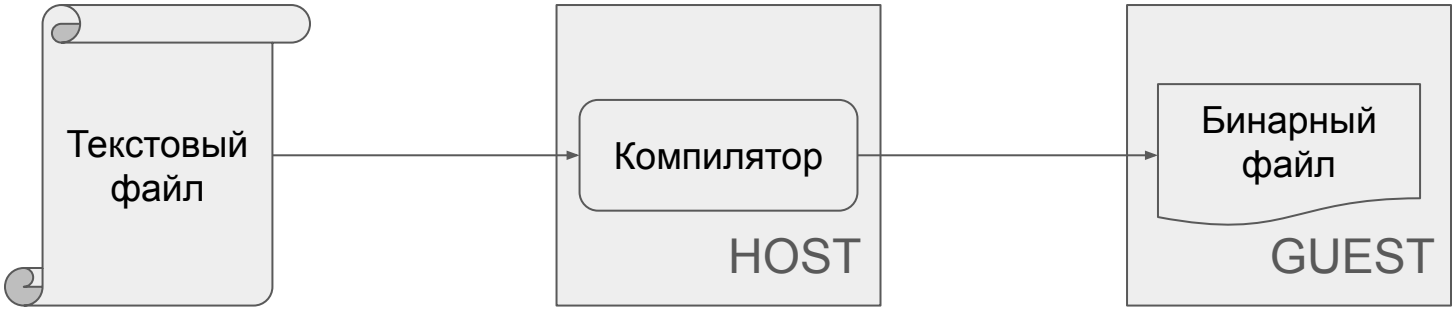
Понятие транслятора

вход/выход	Бинарный файл	Исполнение
Текстовый файл	Компилятор	Интерпретатор
Бинарный файл	Бинарный транслятор	Симулятор

Понятие транслятора

вход/выход	Бинарный файл	Исполнение
Текстовый файл	Компилятор	Интерпретатор
Бинарный файл	Бинарный транслятор	Симулятор

Кросс-компиляция



- HOST = GUEST - нативная компиляция
- HOST ≠ GUEST - кросс-компиляция

Цели курса

- Получить практический опыт реализации функционального симулятора согласно требованиям спецификации архитектуры системы команд
- Освоение промышленных методов разработки, тестирования, отладки и изучения поведения программ
- Опыт работы в команде

Беседа курса