# Слайд 1

**Здравствуйте, уважаемые слушатели! Меня зовут** Переведенцев Артем. **Название моей работы: “**Разработка языка собственного программирования ”. Моими консультантами были **Данила** **Байгушев** и **Никита** **Уваров** – выпускники нашей школы.

# Слайд 2

Целью моей работы было – **разработать** **систему** **трансляции** **программ** **на** **C**-**подобном** **языке** **и** **их** **исполнения**.

В нее входило несколько задач:

**Компилятор, Ассмеблер, Процессор**.

Каждая из этих задач имела свои подзадачи.

В компилятор входила разработка **лексического** **анализатора**, **синтаксического** **анализатора**, **семантического**

**анализатора** и **т.д**.

1

В ассемблер входила разработка системы трансляции команд в машинный код.

В процессор – исполнение машинного кода.

Обо всем этом подробнее в следующих слайдах.

# Слайд 3

**Давайте сначала разберемся, как же заветные строчки кода превращаются в команды для процессора**.

Как и в любом языке программирования пользователь сначала пишет программу. Затем модуль **“Compiler”** начинает работать с кодом.

Первый этап это **лексический анализатор.** На входе он получает файл с программой, на выходе выдает поток лексем.

Следующий этап это **препроцессор**. На входе он получает поток лексем, на выходе выдает обработанный поток лексем.

Далее идет **синтаксический анализатор.** На входе он получает поток лексем, на выходе выдает дерево с программой.

2

Следующими идут два этапа: **семантический анализатор** и **оптимизатор.** Их работу мы разбирать не будем, **т.к. они не влияют на итоговый результат**.

И последний этап это **кодогенератор**. На входе он получает дерево с программой, на выходе выдает файл с командами для ассемблера.

Затем мы переходим к следующему модулю - **“Assembler”.**

Этот модуль работает только с ассемблерным кодом и его результатом является файл с командами для процессора (байткодом).

# Слайд 4

**Начнем с самого первого этапа**. Это лексический анализатор.

Его задача это разбиение текста программы на лексемы или, иначе говоря, парсинг.

Лексема может быть нескольких типов: **число**, **ключевое** **слово**, **команда** **для** **препроцессора**, **оператор**, **имя**.

3

# Слайд 5

Следующий слайд это синтаксический анализатор. **Это самая важная часть компилятора**. На данной стадии происходит построение дерева (AST) по потоку лексем, с помощью особых

грамматических правил методом рекурсивного спуска. **Некоторая часть правил представлена на слайде**.

# Слайд 6

После построения дерева мы переходим к кодогенерации. Операции в ассемблере делятся на несколько типов: арифметические, работа с памятью, переходы, другие.

Результатом работы этого анализатора является обход дерева и запись команд в файл с ассемблерным кодом.

# Слайд 7

**После того, как программа собрана, мы должны ее на чем-то запустить**. Для этого была разработана виртуальная машина, исполняющая команды.

4

Как и в настоящем процессоре в ней имеются: **оперативная память (RAM), набор регистров, два стека и флаги**. Затем происходит исполнение команд.

# Слайд 8

Перейдем к результатам. На слайде вы видите пример простой программы. В данном случае был выбран алгоритм Евклида.

Так же вы можете видеть рядом результат построения дерева из программы. Некоторые вершины в дереве выделены серым цветом. Эти вершины являются служебными и нужны только для построения дерева.

**В ходе работы было написано** приблизительно 5000 строк кода в 24 файлах. На выполнение работы было затрачено около 350 часов.

# Слайд 9

Перейдем к итогам.

5

В процессе выполнения работы были реализованы следующие задачи разработка: **лексического** **анализатора**, **синтаксического** **анализатора**, **семантического**

**анализатора** и **т.д**. (обвести указкой область на слайде). **Также были получены неоценимые знания по работе с деревьями**.

# Слайд 10

Всем спасибо за внимание! Вопросы?

6