

КОМПИЛЯТОР СТЕКОВОГО ЯЗЫКА ПРОГРАММИРОВАНИЯ

Залыгин Вячеслав, ИУ6-73Б

ВВЕДЕНИЕ

Решаемая в рамках ВКРБ задача: разработка компилятора стекового языка программирования.

Проблема: компилятор – сложная система с множеством взаимодействующих компонентов (лексический, синтаксический анализ, оптимизация, генерация кода).

Роль ТСиСА: методологическая основа для анализа, проектирования и оптимизации такой системы.

ЦЕЛЬ И ЗАДАЧИ ВКРБ

Цель (системная): Создать корректно работающую, оптимизированную систему (компилятор), удовлетворяющую заданным требованиям.

Задачи, решаемые методами ТСиСА:

- Анализ: Определение границ, входов, выходов и функций системы (IDEF0).
- Проектирование: Декомпозиция на модули, проектирование взаимодействий.
- Оптимизация: Выявление и устранение узких мест в процессах компиляции.
- Верификация: Обеспечение логической согласованности и надежности системы.

ОСНОВНЫЕ ПРИНЦИПЫ

Система: Компилятор рассматривается как целостный объект с четкими границами, входами (исходный код) и выходами (исполняемый код/байткод).

Структурный подход: Компилятор декомпозируется на взаимосвязанные функциональные подсистемы.

Целенаправленность: Каждый модуль компилятора имеет конкретную цель в рамках общей задачи трансляции.

Связность: Важно не только спроектировать модули, но и четко определить интерфейсы и потоки данных между ними.

КЛАССИФИКАЦИЯ СИСТЕМЫ

Искусственная, техническая и информационная система.

Открытая: зависит от среды (версии библиотек, импорта функций окружения, параметров компиляции).

Сложная и иерархическая: несколько уровней абстракции (текст → AST → IR → целевой формат).

Динамическая: для разных входов и флагов компиляции поведение существенно различается.

Система с качественными показателями: корректность, производительность, переносимость, сопровождаемость (их нужно “прикреплять” к подсистемам через метрики и тесты).

ОБЪЕКТ ИССЛЕДОВАНИЯ: СТЕКОВЫЕ ЯЗЫКИ ПРОГРАММИРОВАНИЯ

Особенность: исполнение основано на стековой виртуальной машине (Forth, Joy, Factor, WebAssembly).

Проблематика компилятора: эффективная трансляция кода, оптимизация стековых операций, управление памятью.

Результат НИР: проведен сравнительный анализ языков (синтаксис, типизация, оптимизации, управление памятью).

ПРИМЕНЕНИЕ IDEFo: ОПРЕДЕЛЕНИЕ КОНТЕКСТА СИСТЕМЫ

Контекстная диаграмма (A-0): Определяет компилятор как «черный ящик».

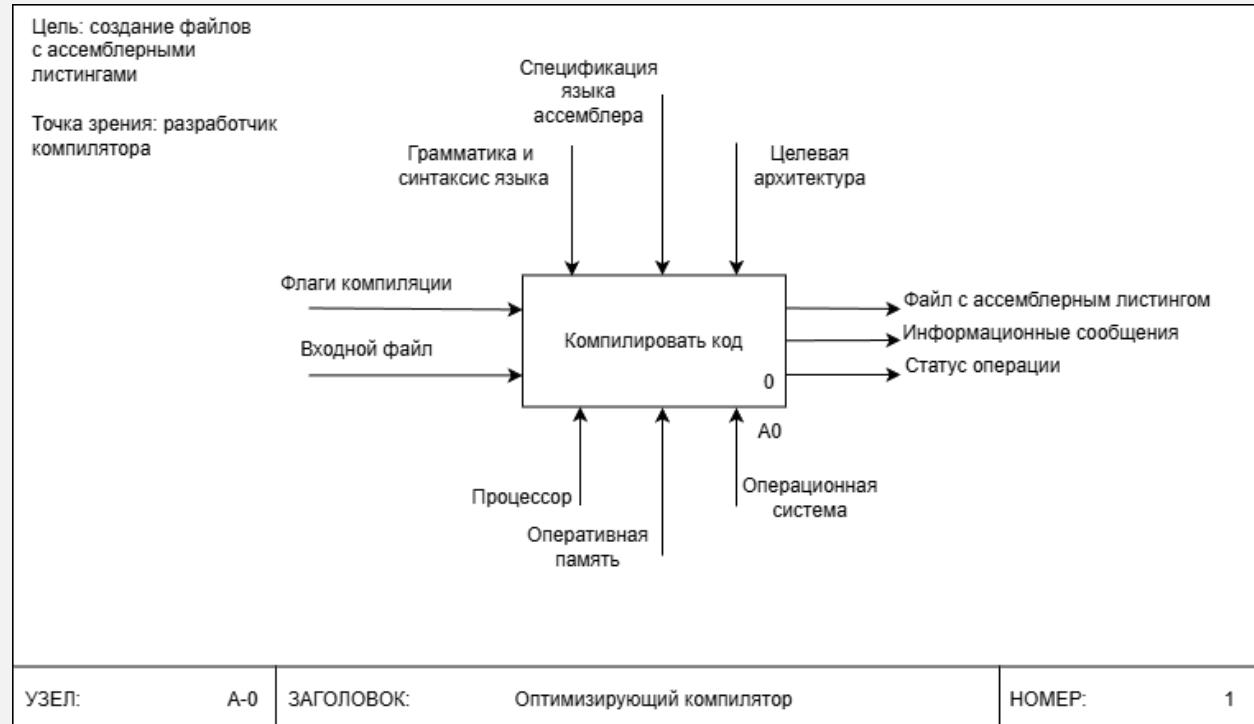
Входы: Исходный код, спецификации языка.

Выходы: Целевой код (ассемблер/байткод), сообщения об ошибках.

Управление: Грамматика языка, флаги компиляции.

Механизмы: Процессор, память, разработчик.

Результат: Четкое понимание границ системы и ее взаимодействия с внешним миром.



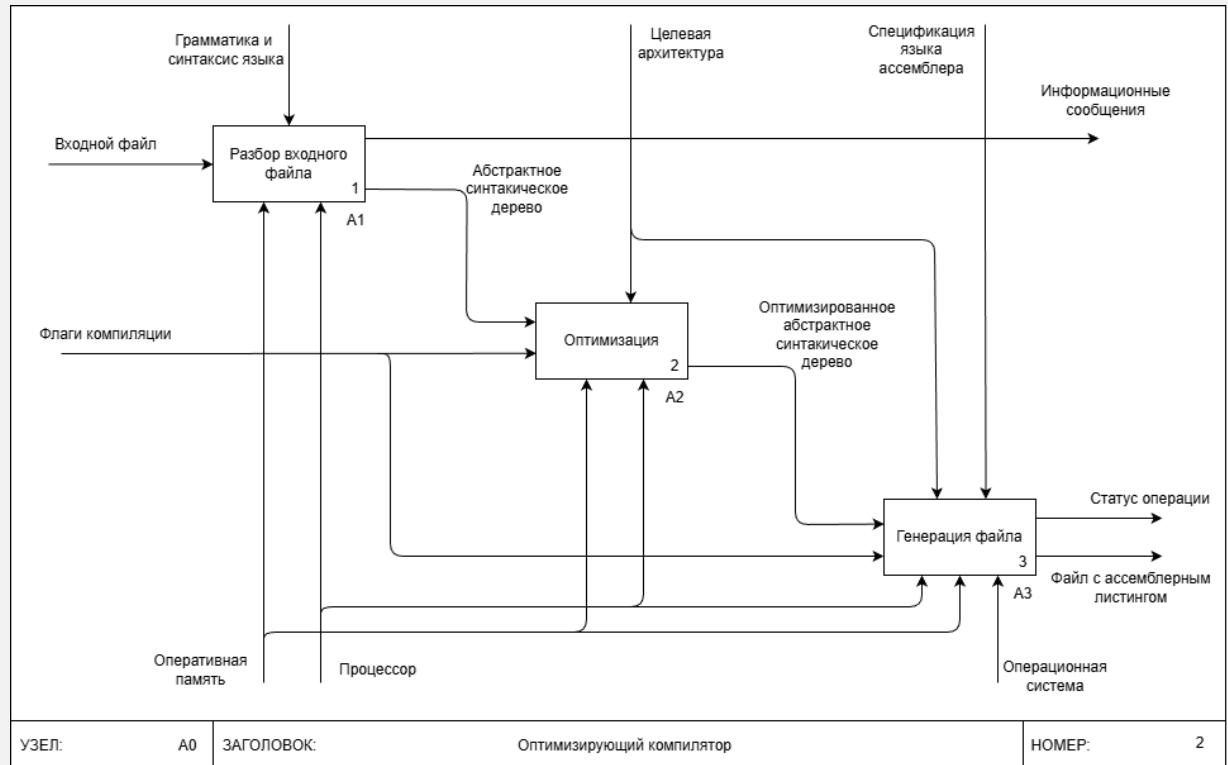
МОДЕЛИРОВАНИЕ ПРОЦЕССОВ КОМПИЛЯЦИИ

Декомпозиция главной функции: «Компилировать код».

Ключевые подпроцессы:

- А1: Разбор входного файла (лексический, синтаксический анализ).
- А2: Оптимизация (локальная, глобальная).
- А3: Генерация выходного файла.

Наглядное представление основных этапов работы компилятора и потоков данных между ними (АСТ, промежуточные представления).



ДЕТАЛЬНЫЙ АНАЛИЗ ПОДПРОЦЕССОВ

В рамках процесса компиляции задействованы следующие подпроцессы:

- А1 «Разбор»: Детализация на лексический анализ и построение АСТ. Выявлено: Отсутствие проверки корректности входных данных.
- А2 «Оптимизация»: Декомпозиция на локальные и глобальные оптимизации. Выявлено: Нет обратной связи для многопроходной оптимизации.
- А3 «Генерация»: Процесс преобразования АСТ в целевой код.

Итог: методология IDEF0 позволила структурировать функциональные требования к каждому модулю.

ВЫЯВЛЕНИЕ И УСТРАНЕНИЕ УЗКИХ МЕСТ (АНАЛИЗ AS-IS И ТО-BE)

Критический анализ существующей (или проектируемой) модели для выявления проблем.

Узкое место (AS-IS)

- Нет проверки корректности входного файла →
- Оптимизация однопроходная, не учитывает синергию

Способ исправления (ТО-BE)

- Добавить блок проверки в процесс разбора (A1)
- Ввести обратные связи между блоками оптимизации (A2, A21)

ПЕРЕХОД ОТ МОДЕЛИ AS-IS К TO-BE

Модель AS-IS: Фиксирует начальное понимание системы, выявляет недостатки.

Модель ТО-BE: Целевая, улучшенная архитектура компилятора.

Что улучшено (по итогам анализа):

- Добавлена валидация входных данных.
- Усложнена и улучшена схема оптимизаций за счет обратных связей.
- Повышена связность и управляемость процессов.

Результат: Проектная документация для реализации.

СТРУКТУРНЫЙ ПОДХОД И ДЕКОМПОЗИЦИЯ В ПРОЕКТИРОВАНИИ

Какие преимущества получены для ВКРБ:

- Упрощение сложности: Задача «написать компилятор» разбита на подзадачи «реализовать лексер», «реализовать оптимизатор» и т.д.
- Четкие интерфейсы: Потоки данных между модулями определены на диаграммах.
- Локализация ошибок: Проблемы проще искать в конкретном модуле.
- Упрощение тестирования: Возможность тестировать модули по отдельности.

СРАВНИТЕЛЬНЫЙ АНАЛИЗ КАК ИНСТРУМЕНТ СИСТЕМНОГО ВЫБОРА

Задача: Выбор решений для архитектуры компилятора.

Примеры решений на основе анализа:

- Типизация: Использовать статическую (как в Cat/Wasm) или динамическую (как в Factor) модель?
- Оптимизации: Применять ли технику «stack caching» или «нитевой код»?
- Управление памятью: Интегрировать сборщик мусора (как Factor) или делегировать языку-источнику (как Wasm)?

Роль ТСиСА: Обеспечение системного, а не хаотичного выбора на основе классификации и сравнения аналогов.



ПРАКТИЧЕСКИЕ РЕЗУЛЬТАТЫ ПРИМЕНЕНИЯ ТСИСА В ВКРБ

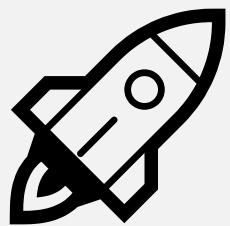
Формализованная архитектура: есть четкая диаграмма модулей компилятора и их взаимодействия.

Список требований: сформирован на основе моделей AS-IS/TO-BE.

План реализации: декомпозиция дает четкий план разработки.

Критерии проверки: понимание того, как должна работать система в целом и ее части.

Основа для технического задания: все ключевые аспекты системы задокументированы.



ВЫВОДЫ

ТСиСА предоставила строгий методологический аппарат для работы над сложным проектом (компилятором).

Методология IDEFo - эффективный инструмент для функционального моделирования и проектирования архитектуры.

Подход AS-IS/TO-BE позволил перейти от анализа проблем к проектированию улучшенного решения.

Структурный подход и декомпозиция — ключ к управлению сложностью в инженерных проектах.

Применение ТСиСА на всех этапах (НИР, проектирование, реализация) повышает системность, обоснованность и качество работы.

