Документация

Проект представляет собой набор из 6 компонентов:

1. Структура ассемблера
2. Вычислитель
3. Компиляторы для инициализации вычислителя и компилирования ассемблера
4. Контроллер, который управляет всей системой
5. IDE
6. Unit tests

Все они, кроме IDE, написаны на языке F#. IDE написана на C#.

Ассемблер

Структура ассемблера реализована в виде Discriminated Union.

Ассемблер поддерживает как однопоточные, так и многопоточные программы.

Ассемблер состоит из 4 команд:

1. Eps, которая ничего не делает и нужна для заполнения места при написании многопоточного кода
2. Set ((row, col), value), которая устанавливает значение value в ячейку (row, col)
3. Mvc ((row, col), value), которая применяет value к ячейке (row, col)
4. Mov ((rowTo, colTo), (rowFrom, colFrom)), которая применяет значение из ячейки (rowFrom, colFrom) к ячейке (rowTo, colTo)

Вычислитель (Processor<'T>)

Вычислитель представляет собой виртуальный процессор архитектуры TTA. Он состоит из ячеек, расположенных в виде матрицы размером бесконечность на N. Поддерживаются любые типы. Ячейки реализованы в виде класса Grid<'T>, который содержит методы для чтения, записи и выполнения функции над ячейкой, а также позволяет узнать размеры (как реальные, зависящие от записанных ячеек, так и максимально возможные).

Вычислитель инициализируется массивом двухместных функций типа 'T -> 'T -> 'T. Каждая функция соответствует отдельному столбцу в матрице.

Ячейки создаются при записи значения в них. При чтении из несуществующей ячейки берется стандартное значение для данного типа.

Каждому столбцу соответствует некоторая бинарная функция, которая применяется к содержимому ячейки (левый операнд) и последнему аргументу команд Mvc (непосредственно) и Mov (после чтения значения из соответствующей ячейки) (правый операнд).

При обращении к ячейкам, которые находятся за пределами вычислителя, возникает Exception.

При выполнении параллельного кода сначала происходит чтение всех ячеек, а затем запись. Таким образом, в группе команд, выполняемых параллельно, возможно сколько угодно чтений из одной и той

же ячейки, но недопустима запись в одну и ту же ячейку. При попытке запустить такой код возникает Exception.

Вычислитель также позволяет выполнить проверку ассемблера и вывести все ошибки, содержащиеся в нем.

Основные методы и свойства класса Processor<'T>	
Read	Читает значение из ячейки.
ReadAll	Возвращает все записанные ячейки.
Evaluate	Вычисляет ассемблер. Есть перегрузки для одной команды, строки и программы. При обнаружении ошибки возникает Exception.
Check	Проверяет ассемблер на ошибки. Есть перегрузки для одной команды, строки и программы.
Clear	Очищает матрицу.
Size	Количество функций.
Width, Height	Реальные размеры вычислителя.

Компилятор ассемблера (AsmYaccCompiler<'T>)

Для синтаксического анализа кода используется YaccConstuctor.

Код хранится в виде массива строк (каждая строка соответствует отдельному потоку). Синтаксический анализатор (парсер) получает на вход код и возвращает либо структурное представление ассемблера, либо массив ошибок (здесь выводятся только лексические и синтаксические ошибки).

В случае, если ошибок не произошло, другой модуль, Контроллер, проверяет ассемблер на на ошибки с помощью вычислителя, что позволяет выявить логические ошибки в структуре кода (несколько записей в одну ячейку, обращение к несуществующей ячейке).

Компилятор реализует единственный метод интерфейса IAsmCompiler - Compile.

Также в коде содержится вторая версия компилятора, основанная на регулярных выражениях, но итоговая программа ее не использует.

Контроллер (Controller<'T>)

Контроллер управляет работой всего приложения: создает, открывает, сохраняет проекты; инициализирует и конфигурирует вычислитель; компилирует, проверяет на ошибки и запускает код.

Класс Controller<'T> реализует интерфейс IController<'T>.

Основные методы и свойства интерфейса IController<'T>	
New, Open, Save	Соответственно, создают, открывают, сохраняют проект
Init	Инициализирует вычислитель
Compile	Компилирует и проверяет код на ошибки
CompilationErrors	Содержит ошибки, произошедшие во время компиляции (если есть)
Run	Запускает вычисление ассемблера, если нет ошибок
StartDebug, Step, StopDebug	Управляют отладочным режимом
Clear	Очищает матрицу вычислителя
Read, ReadAll	Читают данные из вычислителя

IDE

IDE представляет собой приложение Windows.Forms.

Содержит меню, поля ввода кода, отображение матрицы вычислителя и списка ошибок компиляции.

В реализации используются Reactive Extensions для работы с моделью событий.

IDE позволяет: создавать, сохранять, открывать проекты; настраивать вычислитель; компилировать код и выводить ошибки (двойной щелчок по ошибке выделяет в редакторе предполагаемое место ее возникновения); запускать программу, в том числе и в отладочном режиме (построчно).

IDE поддерживает некоторые комбинации клавиш, используемые в Visual Studio (Ctrl+Shift+B, F5, Ctrl+F5, Shift+F5, F10).

IDE позволяет писать параллельный код. Для этого необходимо установить количество потоков (Settings -> Threads), после чего написать код в появившихся элементах TextBox (каждый TextBox содержит код, который будет выполняться в отдельном потоке). Синхронизация выполняется покомандно, а не построчно, то есть строка может содержать несколько команд, но тогда будут некорректно подсвечиваться ошибки. Если код в разных потоках имеет разную длину с точки зрения количества команд, то он автоматически дополняется командами Eps до достижения равной длины потоков.

Отключенные пункты меню оставлены для удобства дальнейшей разработки.

Unit Tests

Юнит-тесты используются для тестирования компонентов.

Тестируются вычислитель и компилятор.

Для написания юнит-тестов используется фреймворк NUnit.