

АННОТАЦИЯ

В данной пояснительной записке приводится описание разработки ассемблера и генератора детализированных листингов для микропроцессора Intel 8080 в виде интегрированного с текстовым процессором ComText решения и составленного на языке программирования Lua. ПО предназначено для облегчения разработки и документации программ, написанных для учебных микропроцессорных стендов УМК-80. Работа также преследует побочную цель демонстрации возможностей текстового процессора ComText как средства составления сложной документации, в том числе с применением автоматизированной вёрстки.

Процесс разработки следует стандартной схеме и включает изучение языка ассемблера микропроцессоров Intel 8080, реализацию парсера, ассемблера и инструментов форматирования для генерации листинга. Для изучения языка ассемблера применялось выпущенное в 1977 году официальное руководство. Для составления парсера была применена библиотека LPEG для составления парсеров на основе РВ-грамматики, включённая в доступные по умолчанию в текстовом процессоре ComText библиотеки. Демонстративные инструменты форматирования используют интерфейс для взаимодействия с ComText через Lua, но в будущем предполагается интерфейс, более приближенный к самому текстовому процессору, и как следствие более удобный.

ComText является весьма непопулярным текстовым процессором. Это можно объяснить отсутствием маркетинга, необходимостью составления документов на специальном (предметно-ориентированном) языке программирования, усиленным фокусом на качество вёрстки ценой сложности инструмента. В связи с этим возможности внедрения разработанного ассемблера практически отсутствуют, пока требования к качеству вёрстки в академической среде ограничены возможностями более популярных текстовых процессоров, прежде всего Microsoft Office Word.

Полученное на момент написания пояснительной записки программное решение имеет функционал, достаточный для использования его в указанных целях, но требует доработки в полноценный модуль для текстового процессора для более удобной работы. В частности, нужно реализовать поддержку ассемблерных директив, составления программ отдельными сегментами, а также предоставить возможность гибкой настройки вида листинга.

ANNOTATION

This explanatory note contains a description of the development process of an assembler and code listing generator for the Intel 8080 line of microprocessors in the form of a solution integrated with the `ContExt` text processor and written in the Lua programming language. This software is intended to simplify the development and documentation of programmes written for the `UMPK-80` educational microprocessor system. This work also intends to demonstrate the typesetting capabilities of `ContExt` in the realm of documentation, including the usage of its automated typesetting instruments.

The development process follows standard procedure and includes studying the assembly language itself followed by developing an appropriate parser, assembler and formatter capable of typesetting a code listing. The official Intel 8080 assembly language manual is the main reference used. The parser has been written using the `LPEG` library for writing parsers based on parsing expression grammars. It comes by default with `ContExt`. A demonstrative formatter has been developed that depends mainly on an `API` that allows using `ContExt` functionality from Lua, however the final product will include a formatter that is more integrated with the text processor itself.

`ContExt` is largely unpopular. It may be caused by a lack of marketing, usage of a domain oriented language being a requirement for writing documents, the complexity of the tool coming as a price for high quality typesetting. It comes as no surprise that there is little hope for integrating the developed software, not until academic typesetting requirements grow beyond the possibilities of much more popular text processing software such as Microsoft Office Word.

At the moment of writing this explanatory note the developed software solution is capable of achieving the given goals, however it requires further development to make it easier to use. It is planned that the solution will receive assembly directive support, literate programming style source code splitting and more flexible code listing typesetting capabilities.

ОБОЗНАЧЕНИЯ И СОКРАЩЕНИЯ

- ConT_{EX}t — текстовый процессор на основе системы компьютерной вёрстки T_{EX}. [1] Разрабатывается нидерландской компанией PRAGMA ADE, занимающейся автоматизированной высококачественной вёрсткой. Использует собственно разработанную новейшую версию T_{EX} под названием LuaMetat_{EX}, имеющую множество улучшений, в том числе встроенный интерпретатор Lua. [2]
- Intel 8080 — второй восьмибитный микропроцессор от Intel, выпуск которого начался в апреле 1974 года.
- LPEG — Lua Parsing Expression Grammar, библиотека для составления парсеров на основе РВ-грамматики в Lua. [3]
- Lua — легко встраиваемый и расширяемый интерпретируемый язык программирования. Нередко используется в полноценных программных решениях как способ пользователю расширить функционал программы. Отличается простотой, скоростью работы интерпретатора и широким применением хеш-таблиц. [4]
- РВ-грамматика — грамматика, разбирающая выражение. Один из типов формальной грамматики. Применяется для обработки компьютерных языков (см. также контекстно-свободную грамматику). [5]
- умПК-80 — учебно-методический стенд, предназначенный для практического изучения микропроцессорных систем на начальном уровне. В основе лежит микропроцессор Intel 8080. [6] Используется в Сургутском государственном университете преподавателями и студентами на направлениях «Управление в технических системах» и «Программная инженерия».

СОДЕРЖАНИЕ

Содержание	4
Введение	5
Основная часть	7
Заключение	8
Список использованных источников	9
Приложение А Характеристика текстового процессора ConT _{Ext}	10

ВВЕДЕНИЕ

Выполнение лабораторных работ с учебно-методическим стендом УМК-80 включает в себя анализ проблемы, проектирование решения, последующую реализацию и тестирование с возможными доработками по мере необходимости. Это проводится в целях закрепления учебного материала по дисциплине «Организация ЭВМ». [6] Далее необходимо привести отчёт, одним из обязательных элементов которого является детальный листинг в виде таблицы, включающей в себя адреса каждого байта машинного кода программы, метки для наглядности, сам машинный код и эквивалентный ассемблерный код, а также комментарии (таблица 1).

Таблица 1 — Пример листинга

Адрес	Метка	Машинный код	Ассемблерный код	Комментарии
0990H	memcpu:	1A	LDAX D	Байт считывается из источника...
0991H		77	MOV M, A	... и он записывается по адресу назначения
0992H		13	INX D	Следующая ячейка памяти источника...
0993H		23	INX H	Следующая ячейка памяти назначения...
0994H		0D	DCR C	Одним байтом меньше, декремент счётчика
0995H		C2	JNZ memcpu	Если есть ещё байты, продолжить копирование
0996H		90		
0997H		09		
0998H		C9	RET	Выход из подпрограммы

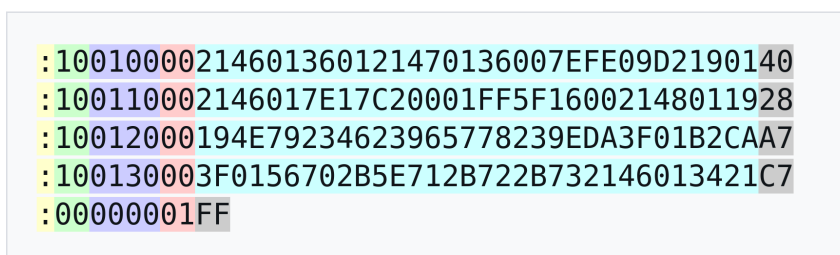
Проблема составления листинга является актуальной, но второстепенной — размер программ, необходимых для выполнения лабораторных работ, достаточно мал, что можно вручную выполнить ассемблирование. В связи с этим данная работа выполняется больше как демонстрация возможностей текстового процессора ComText.

Несмотря на то, что формально листинг необходим лишь как элемент документации, особенности ввода программ на учебном стенде [6] делают его необходимым ещё на этапе реализации и даже реализации решения, прежде всего позволяя сопоставить машинный код и адресное пространство системы для последующего ввода программы в систему. Вариантов составления его не так много:

- Листинг можно составить вручную, что делает большинство студентов. Как правило, для этого используются электронные таблицы (в частности Microsoft Office Excel). При аккуратном ведении листинга можно обеспечить правильную последовательность используемых адресов и даже простоту исправления неверных адресов и

пропущенных команд, но процесс ассемблирования придётся выполнять вручную. По очевидным причинам это несёт за собой вероятность совершения ошибок.

- Можно воспользоваться уже существующими ассемблерами в дополнении к электронным таблицам. В лучшем случае с помощью них можно получить ассемблерный код в виде файла формата Intel HEX (рисунок 1). Как можно увидеть, он является достаточно неудобным для копирования, но действительно упрощает процесс ассемблирования. В частности, можно спутать контрольную сумму с байтом машинного кода программы; приведённые в файле (двухбайтовые) адреса представлены в порядке от старшего байта к младшему в отличие от используемого в микропроцессоре обратного порядка. Сгенерировать файлы этого формата возможно, например, с помощью инструментов эмулятора emuStudio [8]. Есть и онлайн-инструменты, как Pretty Intel 8080 Assembler [9].



The image shows a snippet of Intel HEX format code. Each line represents a record. The fields are color-coded: the first two characters (e.g., '1001') are yellow, the next four characters (e.g., '0002') are purple, the next eight characters (e.g., '214601360121470136007E') are red, the next four characters (e.g., 'FE09D2190140') are cyan, and the last two characters (e.g., 'FF') are grey. The records are separated by line breaks.

```
:10010000214601360121470136007EFE09D2190140
:100110002146017E17C20001FF5F16002148011928
:10012000194E79234623965778239EDA3F01B2CAA7
:100130003F0156702B5E712B722B732146013421C7
:00000001FF
```

Рисунок 1 — Формат Intel HEX. Выделены по порядку: количество байт, адрес, тип записи, данные, контрольная сумма

- В редком случае можно найти инструменты, генерирующие листинг в некоем виде. В частности, в работе «name» [??] представлен эмулятор, имеющий также функционал для генерации листингов в формате, приведённом в таблице 1. Это значительно облегчает составление отчёта, но требуют повторной генерации вне текстовых процессоров в случае изменения программы.

Необходимо отметить, что решения с применением ассемблера предполагают наличие отдельного от текстового процессора ПО. В данной работе представляется решение проблемы, внедрённое в текстовый процессор и позволяющее генерировать листинги напрямую в документе.

ОСНОВНАЯ ЧАСТЬ

ЗАКЛЮЧЕНИЕ

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Атаз-Лопес Х. A not so short introduction to Cont_Ext Mark IV [Электронный ресурс] : неофициальный справочник по Cont_Ext для начинающих / [перевод с испанского анонимным лицом]. — [Б. м.], 2021. — 307 с. — URL: github.com/contextgarden/not-so-short-introduction-to-context (дата обращения 08.06.2025)
2. Хаген Г. Cont_Ext Lua documents [Электронный ресурс] : справочная информация по системе вёрстки LuaMeta_EX — Хасселт: PRAGMA ADE, 2023. — 214 с. — URL: www.pragma-ade.com/general/manuals/cld-mkiv.pdf (дата обращения 08.06.2025)
3. LPEG - Parsing Expression Grammars For Lua [Электронный ресурс] : главная страница проекта LPEG для создания парсеров. — URL: www.inf.puc-rio.br/~roberto/lpeg/ (дата обращения 08.06.2025)
4. Lua: about [Электронный ресурс] : базовые сведения о языке программирования Lua. — URL: lua.org/about.html (дата обращения 08.06.2025)
5. Форд Б. Parsing Expression Grammars: A Recognition-Based Syntactic Foundation [Электронный ресурс] : описание РВ-грамматики. // POPL: Венеция, 2004. — 12 с. — URL: bford.info/pub/lang/peg.pdf (дата обращения 08.06.2025)
6. Запевалов А. В., Запевалова Л. Ю., Степанова Д. П. Методические указания к выполнению лабораторных работ по дисциплинам: «Вычислительные машины, системы и сети», «Организация ЭВМ и систем» // Сургутский государственный университет ХМАО–Югры. — Сургут, иц Сургу, 2020. — 52 с.
7. Intel 8080/8085 Assembly Language Programming : документация по языку ассемблера для микропроцессоров Intel 8080 и 8085. — Intel Corporation, 1978. — 224 с.
8. emustudio.net/
9. github.com/svofski/pretty-8080-assembler
10. Хаген Г. Bibliographies the Cont_Ext Way [Электронный ресурс] : справочная информация по библиографическому модулю Cont_Ext — Хасселт: PRAGMA ADE, 2017. — 102 с. — URL: www.pragma-ade.com/general/manuals/mkiv-publications.pdf (дата обращения 08.06.2025)
11. Хаген Г. Fonts Out of Cont_Ext [Электронный ресурс] : справочная информация по работе со шрифтами в Cont_Ext — Хасселт: PRAGMA ADE, 2016. — 228 с. — URL: www.pragma-ade.com/general/manuals/fonts-mkiv.pdf (дата обращения 08.06.2025)

ПРИЛОЖЕНИЕ А

ХАРАКТЕРИСТИКА ТЕКСТОВОГО ПРОЦЕССОРА `ContEXt`

Высококачественная типография имеет значительную важность для специалистов технических направлений. Одним из актуальных выборов по для качественной вёрстки является пакет программ `ContEXt`.

`ContEXt`—продвинутый текстовый процессор, разрабатываемый компанией PRAGMA Advanced Document Engineering, или PRAGMA ADE. [1] Построенный на новейшей версии системы компьютерной вёрстки `TEX` под названием `LuaMetaTEX` или `LMTX`, `ContEXt` обеспечивает вёрстку проектов-документов высокой сложности благодаря следующим принципам и функционалу:

1. Автоматическое формирование библиографий исходя из содержимого выбранной базы источников (файл) [10];
2. Принцип `wysiwyw` («видишь то, что имеешь в виду») вместо `wysiwyg` («видишь то, что получишь») при составлении документов. Это значит, что пользователь работает не над готовым документом, как в визуальных редакторах (пр. Microsoft Word), а над входным файлом с командами для `ContEXt` и текстом на вывод в документ. [1] Этот файл обрабатывается `ContEXt` и на выход получается файл `PDF`;
3. Семантический подход к элементам документа. Это значит, что, например, раздел—не просто текст, начинающийся с заголовка с определённым стилем шрифта и выравниванием, а логический элемент входного файла, границы которого определяются командами начала и конца раздела. Это делает входной файл более понятным—не нужно знать, как выглядит заголовок раздела, чтобы понять, что раздел начинается с `\startsection`, заканчивается `\stopsection`, а переданный первой команде параметр `[title={Текст заголовка}]` задаёт текст заголовка; [1]
4. Значительная пластичность и очевидная закономерность логических элементов. Для их большинства, имея некий базовый элемент `element`, будут команды создания нового подвида элемента `\defineelement`, настройки элемента в целом или подвида на выбор `\setupelement`, а также обеспечивается использование элемента с помощью одной команды `\element` и/или с помощью команд начала и конца `\startelement` и `\stopelement` соответственно. Это позволяет легко изменить стиль целой группе элементов текста одновременно; [1]
5. Обширная по возможностям настройка шрифта. По умолчанию `ContEXt` следит за четырьмя шрифтами на документ: антиквой (с засечками, основной шрифт для текста), гротеском (без засечек), моноширинным (для кода) и математическим (для формул).

Наборы шрифтов—как этих четырёх, так и дополнительных—называются машинописями. Пользователь способен объявлять собственные машинописи, но есть и заранее готовые, идущие вместе с программой. [11] Позволяется включать любые доступные шрифту особенности (англ. *font features*) в пределах используемого формата шрифта. Так, например, у Times New Roman есть как маюскульные цифры 0123456789, так и минускульные цифры 0ι23456789. Есть и дроби $\frac{0123456789}{0123456789}$. Также поддерживается микротипографика—улучшение читаемости текста за счёт висячих символов и небольшого сжатия/растягивания глифов. [??]

6. Продвинутая математическая вёрстка. ConT_EXt использует всю мощь T_EX для формирования математических конструкций со сложными символами, растущими разграничителями и тому подобное (рисунок А.1);

$$U_2 = \frac{1}{2!} \int_0^\beta d\tau_1 \int_0^\beta d\tau_2 \sum_{\substack{k_1, q_1 \\ k_2, q_2}} \left\langle \mathcal{T} \left[c_{k_1}^\dagger(\tau_1) \Delta_{k_1, q_1}^r c_{-k_1}^*(\tau_1) + c_{-q_1}^T(\tau_1) \Delta_{k_1, q_1}^{r\dagger} c_{q_1}(\tau_1) \right] \right. \\ \left. \times \left[c_{k_2}^\dagger(\tau_2) \Delta_{k_2, q_2}^r c_{-k_2}^*(\tau_2) + c_{-q_2}^T(\tau_2) \Delta_{k_2, q_2}^{r\dagger} c_{q_2}(\tau_2) \right] \right\rangle.$$

Рисунок А.1 — Пример математической вёрстки

7. Автоматическая нумерация разделов, подразделов, рисунков, таблиц и так далее;
8. Использование языка программирования Lua для расширения функционала

и так далее.

Будучи многофункциональным текстовым процессором с возможностью расширения его способностей с помощью легкодоступного языка, решение задачи сборки программы на некоем языке в целях представления её в удобочитаемом виде не представляется сложным.