ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ «СИБИРСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ ТЕЛЕКОММУНИКАЦИЙ И ИНФОРМАТИКИ» КАФЕДРА ВЫЧИСЛИТЕЛЬНЫХ СИСТЕМ ЛАБОРАТОРНАЯ РАБОТА

**Курсовой проект** по дисциплине: Архитектуры ЭВМ

**«Simple Computer»**

Выполнили студенты:

Шпаков Д.С., Монастырный М.Е.

Группа:

ИП-317

Проверила:

Челканова Т.В.

г. Новосибирск, 2025

Оглавление

1. [Полный текст задания 3](#_Toc198143178)

* [Краткое задание 3](#_Toc198143179)
* [Транслятор с языка Simple Assembler 3](#_Toc198143180)
* [L1-кэш команд и данных 4](#_Toc198143181)
* [Транслятор с языка Simple Basic 4](#_Toc198143182)

1. [Реферат 6](#_Toc198143183)

* [Объем ПЗ 6](#_Toc198143184)
* [Краткая характеристика и результаты работы 6](#_Toc198143185)

1. [Содержание 7](#_Toc198143186)

* [Постановка задачи исследования 7](#_Toc198143187)
* [Блок-схемы используемых алгоритмов 7](#_Toc198143188)
* [Результаты проведенного исследования 10](#_Toc198143189)
* [Выводы 10](#_Toc198143190)

1. [Список используемой литературы 11](#_Toc198143191)
2. [Подпись, Дата 12](#_Toc198143192)

# Полный текст задания

Краткое задание:

Разработать транслятор с языка Simple Assembler. Итог работы транслятора – бинарный файл с образом оперативной памяти Simple Computer, который можно загрузить в модель и выполнить;

Доработать модель Simple Computer – реализовать алгоритм работы блока «L1-кэш команд и данных» и модифицировать работу контроллера оперативной памяти и обработчика прерываний таким образом, чтобы учитывался простой процессора при прямом доступе к оперативной памяти;

Разработать транслятор с языка Simple Basiс. Итог работы транслятора – текстовый файл с программой на языке Simple Basic.

Транслятор с языка Simple Assembler

Разработка программ для Simple Computer может осуществляться с использованием низкоуровневого языка Simple Assembler. Для того чтобы программа могла быть обработана Simple Computer необходимо реализовать транслятор, переводящий текст Simple Assembler в бинарный формат, которым может быть считан консолью управления. Пример программы на Simple Assembler: 00 READ 09 ; (Ввод А) 01 READ 10 ; (Ввод В) 02 LOAD 09 ; (Загрузка А в аккумулятор) 03 SUB 10 ; (Отнять В) 04 JNEG 07 ; (Переход на 07, если отрицательное) 05 WRITE 09 ; (Вывод А) 06 HALT 00 ; (Останов) 07 WRITE 10 ; (Вывод В) 08 HALT 00 ; (Останов) 09 = +0000 ; (Переменная А) 10 = +9999 ; (Переменная В) Программа транслируется по строкам, задающим значение одной ячейки памяти. Каждая строка состоит как минимум из трех полей: адрес ячейки памяти, команда (символьное обозначение), операнд. Четвертым полем может быть указан комментарий, который обязательно должен начинаться с символа точка с запятой. Название команд представлено в таблице 1. Дополнительно используется команда =, которая явно задает значение ячейки памяти в формате вывода его на экран консоли (+XXXX). Команда запуска транслятора должна иметь вид: sat файл.sa файл.o, где файл.sa – имя файла, в котором содержится программа на Simple Assembler, файл.o – результат трансляции.

## L1-кэш команд и данных

С целью оптимизации доступа в центральном процессоре используется кэш, в котором хранится 5 строк по 10 значений. Прежде, чем делать запрос к оперативной памяти контроллер памяти проверяет наличие запрашиваемых данных (команд) в кэше. Если они там есть, то запрос выполняется за один такт и значение сразу передается контроллером памяти блоку, который их запрашивал. Если запрашиваемой ячейки нет в кэше, то контроллер запрашивает из оперативной памяти строку (10 значений, выровненных по границе 10) и помещает её в кэш. При этом, если в кэше нет свободных срок, то самая неиспользуемая строка (доступ к ячейкам которой не было дольше всего) выгружается обратно в оперативную память (если она изменялась) и освобождается, а на её место загружается новая строка из оперативной памяти. Для упрощения ситуации считаем, что операция записи выгружаемых данных в оперативную память и чтение новых данных из оперативной памяти осуществляется одним запросом контроллера памяти. При записи значений в оперативную память данные сохраняются в кэше. При этом если соответствующей строки в кэше не было, то она предварительно загружается в него по аналогии с загрузкой данных при запросе из памяти с учетом простоя процессора. Если контроллеру памяти передан некорректный адрес ячейки памяти, то устанавливается флаг выхода за границы памяти и работа контроллера прекращается

Транслятор с языка Simple Basic

Для упрощения программирования пользователю модели Simple Computer должен быть предоставлен транслятор с высокоуровневого языка Simple Basic. Файл, содержащий программу на Simple Basic, преобразуется в файл с кодом Simple Assembler. Затем Simple Assembler-файл транслируется в бинарный формат. В языке Simple Basic используются следующие операторы: rem, input, output, goto, if, let, end. Пример программы на Simple Basic: 10 REM Это комментарий 20 INPUT A 30 INPUT B 40 LET C = A – B 50 IF C < 0 GOTO 20 60 PRINT C 70 END Каждая строка программы состоит из номера строки, оператора Simple Basic и параметров. Номера строк должны следовать в возрастающем порядке. Все команды за исключением команды конца программы могут встречаться в программе многократно. Simple Basic должен оперировать с целыми выражениями, включающими операции +, -, \*, и /. Приоритет операций аналогичен C. Для того чтобы изменить порядок вычисления, можно использовать скобки. Транслятор должен распознавания только букв верхнего регистра, то есть все символы в программе на Simple Basic должны быть набраны в верхнем регистре (символ нижнего регистра приведет к ошибке). Имя переменной может состоять только из одной буквы. Simple Basic оперирует только с целыми значениями переменных, в нем отсутствует объявление переменных, а упоминание переменной автоматически вызывает её объявление и присваивает ей нулевое значение. Синтаксис языка не позволяет выполнять операций со строками.

# Реферат

## Объем ПЗ

Страниц: 12;

Количество таблиц: 0;

Рисунков: 2;

Схем: 2;

Программ: 17;

Приложений: 2;

## Краткая характеристика и результаты работы

В рамках курсовой работы разработан транслятор с языка Simple Assembler, преобразующий программы в бинарный формат. Модель Simple Computer доработана с реализацией L1-кэша, контроллера памяти и обработчика прерываний.

# Содержание

## Постановка задачи исследования

Цель – разработать транслятор с языка Simple Assembler в бинарный формат, а также доработать модель Simple Computer с реализацией L1-кэша, модификацией контроллера оперативной памяти и обработчика прерываний.

## Блок-схемы используемых алгоритмов

Работа транслятора с языка Simple Assembler:

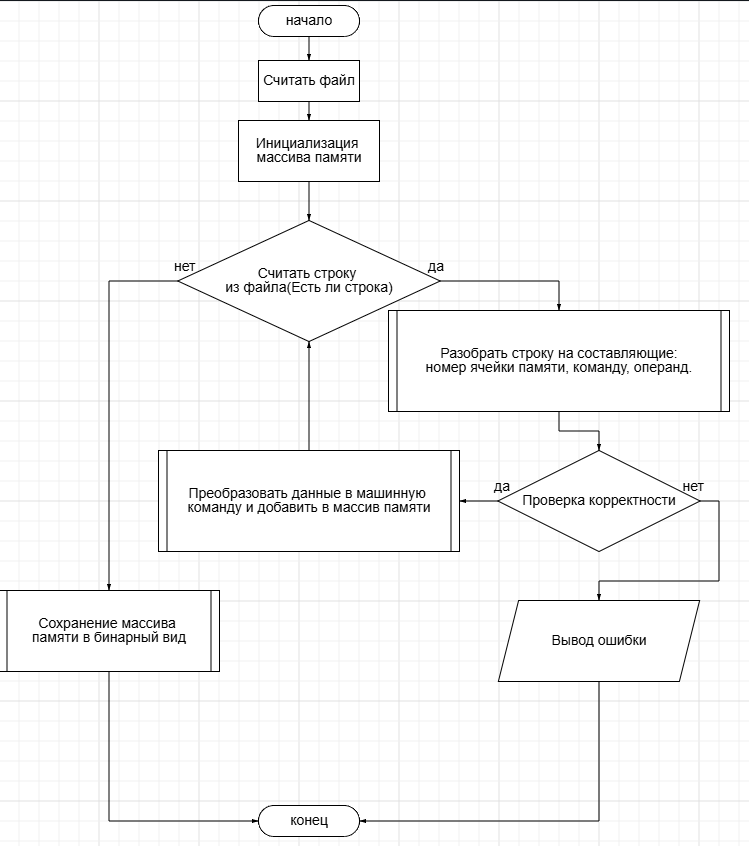


Рисунок 1

Работа Кэша:

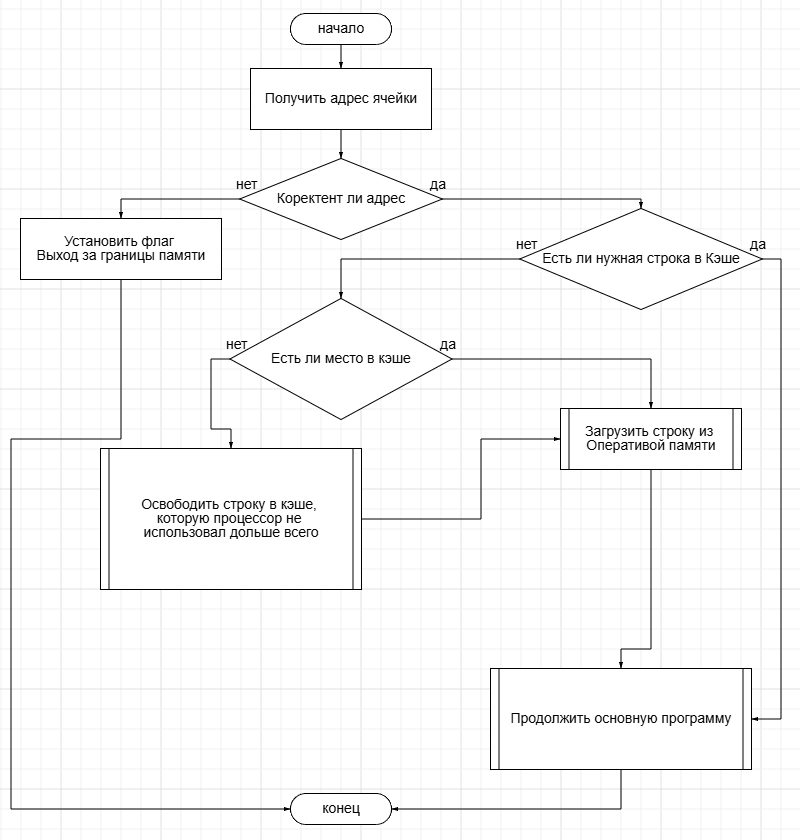


Рисунок 2

**Программная реализация**

Проект реализован на языке C 11 версии. Полная структура:  
simple-computer  
├── console  
│ ├── console.c  
│ └── pr01.c  
├── include  
│ ├── myALU.h  
│ ├── myBigChars.h   
│ ├── myCU.h  
│ ├── myCashe.h  
│ ├── myReadKey.h  
│ ├── mySimpleComputer.h  
│ └── myTerm.h  
├── myALU  
│ └── myALU.c

├── myAssembler  
│ ├── Makefile  
│ ├── [myAssembler.c](https://git.csc.sibsutis.ru/ip317s26/simple-computer/-/blob/rgr/myAssembler/myAssembler.c?ref_type=heads)  
│ └── [program.sa](https://git.csc.sibsutis.ru/ip317s26/simple-computer/-/blob/rgr/myAssembler/program.sa?ref_type=heads)  
├── myBigChars  
│ ├── charpos.c  
│ ├── filechar.c  
│ └── printchar.c  
├── myCU  
│ └── myCU.c   
├── myCashe  
│ └── myCashe.c  
├── myReadKey  
│ └── myReadKey.c  
├── mySimpleComputer  
│ ├── memory.c  
│ ├── out.c  
│ ├── processor.c  
│ └── registr.c  
├── myTerm  
│ ├── color.c  
│ ├── default.c  
│ └── screen.c  
├── .gitignore  
├── .gitlab-ci.yml   
├── CASHE1.dat  
├── DAN1.dat  
├── DAN2.dat  
├── Makefile  
├── MISH1.dat  
├── MISH2.dat  
├── README.md  
├── RUN.dat  
├── test\_terminal.sh  
└── test\_terminal2.sh

Makefile:   
make build - сборка  
make run - запуск  
make clean - чистка временных файлов

## Результаты проведенного исследования

Транслятор корректно переводит Simple Assembler в бинарный формат.

Модель работает с учетом кэширования и задержек.

Проведено тестирование, подтверждающее корректность работы.

## Выводы

В рамках курсовой работы:

* Разработан транслятор Simple Assembler;
* Доработана модель Simple Computer;
* Проведено тестирование;
* Получены удовлетворительные результаты

# Список используемой литературы

1. Методические указания по выполнению курсовой работы
2. Документация по Simple Basic и Simple Assembler
3. Учебные ресурсы по языку C

# Подпись, Дата

Студент: Шпаков Данил Сергеевич ИП-317

Студент: Монастырный Михаил Евгеньевич ИП-317

Дата: 14 мая 2025 года