ФЕДЕРАЛЬНОЕ АГЕНТСТВО СВЯЗИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ

БЮДЖЕТНОЕ

ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО

ОБРАЗОВАНИЯ

«СИБИРСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

ТЕЛЕКОММУНИКАЦИЙ И ИНФОРМАТИКИ»

Кафедра вычислительных систем

КУРСОВАЯ РАБОТА по дисциплине «Архитектура ЭВМ»

Выполнил:

студент группы ИП-

Проверил: доцент кафедры вычислительных систем:

Майданов Юрий Сергеевич Новосибирск, 2024

Оглавление

[Постановка задачи 1](#_Toc87815)

[Блок-схемы использованных алгоритмов 4](#_Toc87816)

[Программная реализация 6](#_Toc87817)

[Результаты проведённого исследования 12](#_Toc87818)

[Выводы 16](#_Toc87819)

[Список использованной литературы 16](#_Toc87820)

[Листинг 16](#_Toc87821)

# Постановка задачи

В рамках курсовой работы необходимо:

* Разработать транслятор с языка Simple Basic. 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:

1. READ 09 ; (Ввод А)
2. READ 10 ; (Ввод В)
3. LOAD 09 ; (Загрузка А в аккумулятор)
4. SUB 10 ; (Отнять В)
5. JNEG 07 ; (Переход на 07, если отрицательное)
6. WRITE 09 ; (Вывод А)
7. HALT 00 ; (Останов)
8. WRITE 10 ; (Вывод В)
9. HALT 00 ; (Останов)
10. = +0000 ; (Переменная А)
11. = +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.

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.

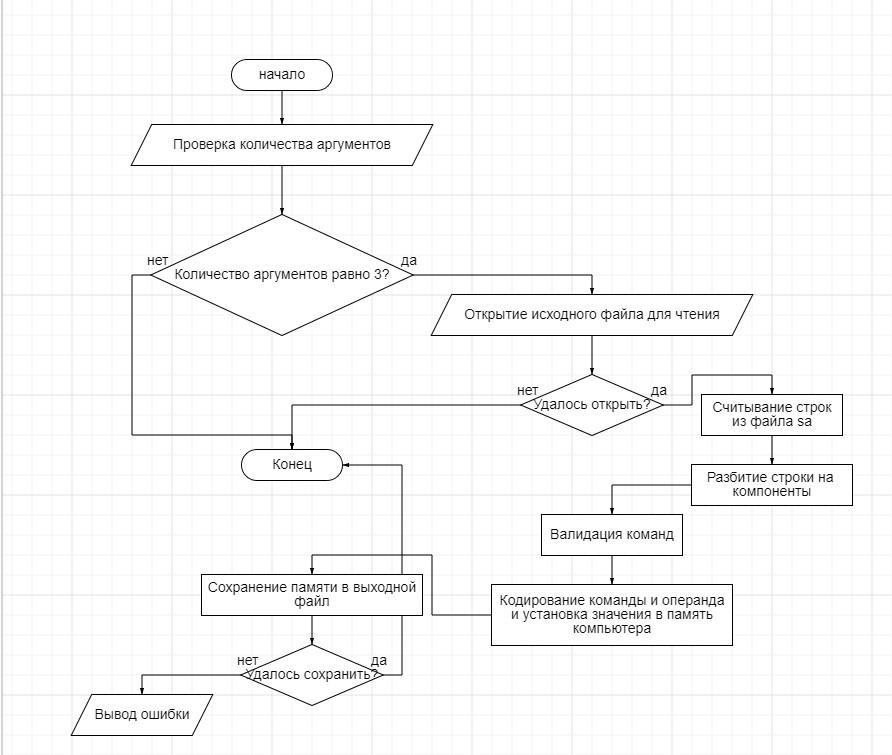
Транслятор должен распознавания только букв верхнего регистра, то есть все символы в программе на Simple Basic должны быть набраны в верхнем регистре (символ нижнего регистра приведет к ошибке). Имя переменной может состоять только из одной буквы. Simple Basic оперирует

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.

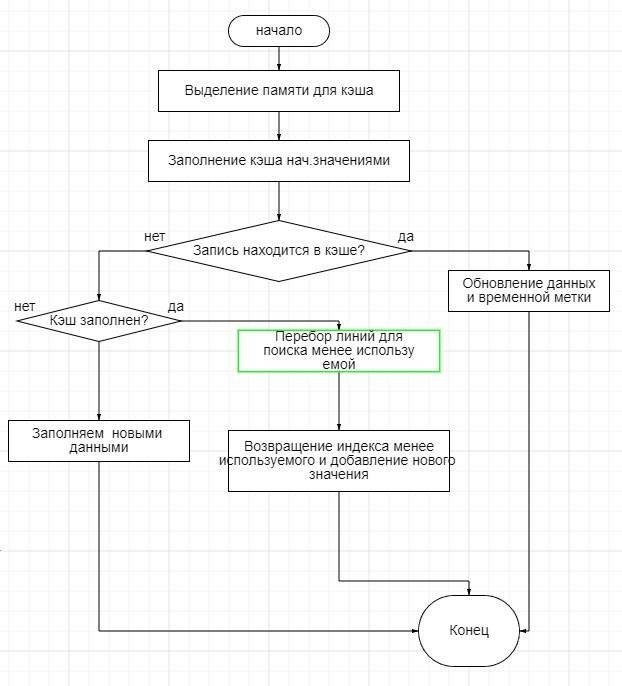
# 

# Блок-схемы использованных алгоритмов

SimpleAssembler:



Кеш процессора:



# Программная реализация

Кеш процессора:

sc\_initCache ():

Функция инициализирует кэш, выделяя память под массив структуру и устанавливает начальные значения для каждой структуры, такие как адрес и время последнего доступа

sc\_findLeastRecentlyUsedCacheEntry ():

Функция проходит через все записи в кэше и ищет индекс наименее недавно использованной записи, основываясь на времени их последнего доступа.

updateCacheLine:

Функция обновляет содержимое одной строки кэша. Она копирует данные из памяти, начиная с указанного адреса address, в соответствующую строку кэша, и обновляет адрес этой строки кэша и время её последнего доступа.

sc\_updateCacheAfterSave:

Функция обновляет кэш, учитывая адреса в памяти и их соответствие строкам в кэше, основываясь на том, есть ли уже запись в кэше для данного адреса, и либо добавляя новую запись, либо обновляя существующую.

Simple Assembler:

isValidRange():

Функция проверяет, находятся ли переданные адрес и значение в допустимом диапазоне, возвращая true, если оба соответствуют установленным ограничениям, иначе возвращает false.

memorySet():

Функция устанавливает значение по указанному адресу в массиве памяти. Одновременно она проверяет, находятся ли переданный адрес и значение в допустимом диапазоне при помощи функции isValidRange(), и если нет, устанавливает соответствующий флаг ошибки и выводит сообщение о выходе за пределы допустимого диапазона.

commandValidate():

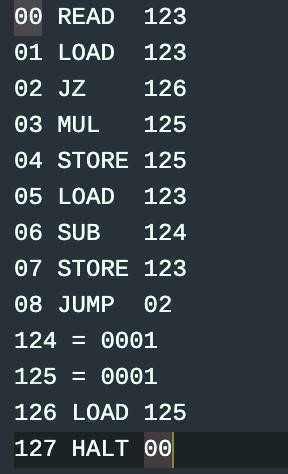
Функция проверяет, является ли переданная строка командой, и если да, возвращает её соответствующий код. Для этого она сравнивает переданную команду с массивом известных команд и их кодов, возвращая соответствующий код, если команда найдена. Если переданная строка начинается с символа '=', она интерпретируется как команда для установки значения, возвращая код 1. Если переданная команда не совпадает ни с одной известной командой, функция возвращает -1.

# Результаты проведённого исследования

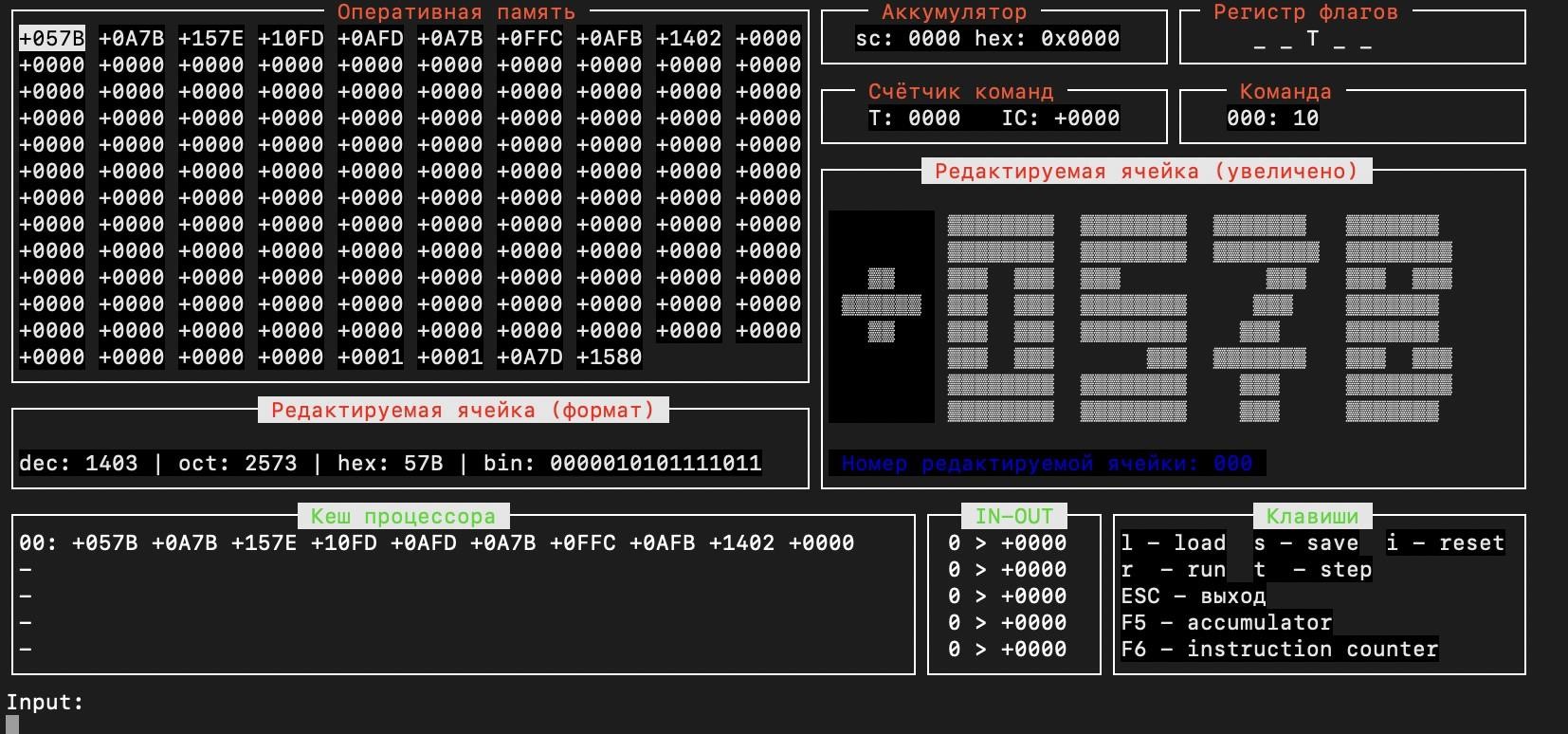
После написания проекта необходимо его протестировать, в качестве теста было предложено написать программу для умножения переменных. Код программы на языке Assembler:

1. READ 123
2. LOAD 123
3. JZ 126
4. MUL 125
5. STORE 125
6. LOAD 123
7. SUB 124
8. STORE 123
9. JUMP 02
10. = 0001
11. = 0001
12. LOAD 125
13. HALT 00

Результат трансляции с SimpleAssembler на В оперативную память:



Результат загрузки файла в SimpleComputer:



Вводим значение A = 5. Ответ = 120

# 

# Выводы

# В рамках курсовой работы была реализована интеграция кэш-процессора, а также разработаны трансляторы с языка Simple Assembler.

# Список использованной литературы

1. ЭВМ и ПЕРИФЕРИЙНЫЕ УСТРОЙСТВА [Учебное пособие]. С.Н.

Мамойленко О.В. Молдованова

1. c-cpp.ru [сайт]. URL: https://www.c-cpp.ru/
2. Язык программирования Си - R-5.

https://www.r-5.org/files/books/computers/languages/c/kr/Brian\_Kernighan\_Dennis\_RitchieThe

\_C\_Programming\_Language-RU.pdf [URL]