Министерство цифрового развития, связи

и массовых коммуникаций Российской Федерации

Федеральное государственное бюджетное образовательное учреждение высшего образования «Сибирский государственный университет телекоммуникаций и информатики» (СибГУТИ)

Кафедра вычислительных систем

**ПОЯСНИТЕЛЬНАЯ ЗАПИСКА**

к курсовой работе

по дисциплине «**Архитектура ЭВМ**»

|  |  |  |
| --- | --- | --- |
| Выполнил:  студент гр. ИС-142  «\_\_» июня 2023 г. | \_\_\_\_\_\_\_\_\_\_\_\_ | /Григорьев Ю.В./ |
|  |  |  |
| Проверил:  преподаватель  «\_\_» июня 2023 г. | **\_\_\_\_\_\_\_\_\_\_\_\_** | /Майданов Ю.С./ |

Оценка «\_\_\_\_\_\_\_\_\_\_\_\_\_»

Новосибирск 2023

**ОГЛАВЛЕНИЕ**

[ПОСТАНОВКА ЗАДАЧИ 3](#_Toc35593781)

[БЛОК-СХЕМЫ АЛГОРИТМОВ 6](#_Toc35593782)

[ПРОГРАММНАЯ РЕАЛИЗАЦИЯ 9](#_Toc35593783)

[РЕЗУЛЬТАТЫ РАБОТЫ ПРОГРАММЫ 12](#_Toc35593783)

[ЗАКЛЮЧЕНИЕ 13](#_Toc35593783)

[ИСПОЛЬЗОВАННАЯ ЛИТЕРАТУРА 15](#_Toc35593783)

# **ПОСТАНОВКА ЗАДАЧИ**

В рамках курсовой работы необходимо доработать модель *Simple Computer* так, чтобы она обрабатывала команды, записанные в оперативной памяти. Система команд представлена в таблице 1. Из пользовательских функций необходимо реализовать только одну согласно варианту задания (номеру вашей учетной записи). Для разработки программ требуется создать трансляторы с языков *Simple Assembler* и *Simple Basic*.

**Обработка команд центральным процессором**

Для выполнения программ моделью *Simple Computer* необходимо реализовать две функции:

**int *ALU*** (*int command, int operand*) – реализует алгоритм работы арифметико-логического устройства. Если при выполнении функции возникла ошибка, которая не позволяет дальше выполнять программу, то функция возвращает -1, иначе 0;

**int *CU*** (void) – обеспечивает работу устройства управления. Обработку команд осуществляет устройство управления. Функция *CU* вызывается либо обработчиком сигнала от системного таймера, если не установлен флаг «игнорирование тактовых импульсов», либо при нажатии на клавишу *t*.

Алгоритм работы функции следующий:

1. из оперативной памяти считывается ячейка, адрес которой хранится в регистре *instructionCounter*;  
2. полученное значение декодируется как команда;  
3. если декодирование невозможно, то устанавливаются флаги «указана неверная команда» и «игнорирование тактовых импульсов» (системный таймер можно отключить) и работа функции прекращается.  
4. Если получена арифметическая или логическая операция, то вызывается функция *ALU*, иначе команда выполняется самим устройством управления.  
5. Определяется, какая команда должна быть выполнена следующей и адрес её ячейки памяти заносится в регистр *instructionCounter*.  
6. Работа функции завершается.

**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.

**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 еѐ объявление и присваивает ей нулевое значение. Синтаксис языка не позволяет выполнять операций со строками.

**Оформление отчета по курсовой работе**

Отчет о курсовой работе представляется в виде пояснительной записки (ПЗ), к которой прилагается разработанное программное обеспечение. В пояснительную записку должны входить:

• титульный лист;  
• полный текст задания к курсовой работе;  
• реферат (объем ПЗ, количество таблиц, рисунков, схем, программ, приложений, краткая характеристика и результаты работы);  
• содержание:  
• постановка задачи исследования;  
• блок-схемы используемых алгоритмов;  
• программная реализация;  
• результаты проведенного исследования;  
• выводы;  
• список использованной литературы;  
• подпись, дата.

Пояснительная записка должна быть оформлена на листах формата А4, имеющих поля. Все листы следует сброшюровать и пронумеровать.

**БЛОК-СХЕМЫ АЛГОРИТМОВ**

1. CU()

Считывание ячейки памяти, номер которой обозначен в instructionCounter

Декодирование полученного значения в команду и операнд

Нет

Команда и операнд корректны?

Да

Выполнение команды управления / вызов ALU() для выполнения арифметической операции

Обновление интерфейса mySimpleComputer

Вывод сообщения об ошибке, установка флагов IGNORE, WRONG\_COMMAND / WRONG\_OPERAND

**2. Транслятор с Simple Assembler**

Получение имён входного и выходного файлов

Входной файл существует?

**Да**

**Нет**

Считывание строки из входного файла

Строка корректна?

**Нет**

**Да**

Кодирование команды, запись в память

Это последняя строка входного файла?

**­­**

**Нет**

**Да**

Вывод сообщения об ошибке

Запись полученного в выходной файл

**3.** Транслятор с Simple Basic

Входной файл существует?

Получение имён входного и выходного файлов

Нет

Да

Считывание строки из входного файла

Строка корректна? (проверка номера строки, команды, операнда)

Нет

Да

Обработка команды: проверка используемых переменных на корректность, получение адресов переменных, выполнение операции (в случае LET – транслирование выражения в RPN, выполнение RPN с помощью стек-машины)

Запись оттранслированного кода команды на Simple Assembler в выходной файл

Это последняя строка входного файла?

Нет

Вывод сообщения об ошибке

Да

# **ПРОГРАММНАЯ РЕАЛИЗАЦИЯ**

# **РЕЗУЛЬТАТЫ РАБОТЫ ПРОГРАММЫ**

# **ЗАКЛЮЧЕНИЕ**

В рамках выполнения курсовой работы мной была доработана модель *Simple Computer* так, чтобы она обрабатывала команды, записанные в оперативной памяти с помощью функций *ALU()* и *CU()*. Также были разработаны транслятор с языка программирования *Simple Assembler*, переводящий код с него в бинарный формат, и транслятор с высокоуровнего языка программирования *Simple Basic*, переводящий его в код на Simple Assembler. Из пользовательских функций необходимо реализовать только одну согласно варианту задания (номеру вашей учетной записи).

**ИСПОЛЬЗОВАННАЯ ЛИТЕРАТУРА**

1. Организация ЭВМ и систем. Практикум // С.Н. Мамойленко, Новосибирск: ГОУ ВПО «СибГУТИ», 2005 г.

2. Архитектура компьютера. 4-е изд. // Э. Танненбаум. – СПб.: Издательство «Питер», 2003.

3. Организация ЭВМ. 5-е изд. / К. Хамахер, З. Вранешич, С. Заки. – СПб.: Издательство «Питер», 2003.

4. Цилькер Б.Я., Орлов С.А. Организация ЭВМ и систем: учебник для ВУЗов. – СПб.: Издательство «Питер», 2004.