МИНОБРНАУКИ РОССИИ

Федеральное государственное бюджетное образовательное учреждение высшего образования

**«ЧЕРЕПОВЕЦКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»**

Институт (факультет)

Кафедра

КУРСОВАЯ РАБОТА

по дисциплине Системное программирование

на тему Программирование на языке низкого уровня

Выполнил студент группы

*группа*

направления подготовки (специальности)

*шифр, наименование*

*фамилия, имя, отчество*

Руководитель

*фамилия, имя, отчество*

*должность*

Дата представления работы

« » 20 г. Заключение о допуске к защите

Оценка

,

количество баллов

Подпись преподавателя

Череповец,

*год*

**Аннотация**

В данной курсовой работе рассмотрен процесс написания программы на ассемблере. Проведено изучение и описание предметной области, для заданного задания. Помимо этого, детально рассматривается каждый этап выполнения.

Курсовая работа включает в себя три приложения: техническое задание, код программы и руководство пользователя.

В техническом задании, в основном, говорится о непосредственной разработке программы. Руководство пользователя содержит подробные инструкции, необходимые пользователю.

Оглавление

1. Введение 4

2. Основная часть 5

2.1 Изучение и описание предметной области 5

2.2 Постановка задачи5

2.3 Выбор структур данных для решения поставленной задачи5

2.4 Логическое проектирование6

2.5 Физическое проектирование программного обеспечения8

2.6 Проектирование интерфейса8

2.7 Кодирование8

2.8 Тестирование8

3. Заключение10

4. Источники информации11

5. Приложение 1. Техническое задание12

6. Приложение 2. Текст программы17

7. Приложение 3. Руководство пользователя19

**Введение**

Данная работа посвящена написанию программы на ассемблере.

Целью курсовой работы является разработка программы в соответствии с заданием.

Для осуществления обозначенной цели необходимо:

* проанализировать задание на курсовую работу;
* определить требования к программному продукту;
* выбрать типы данных, необходимые для решения задачи;
* изучить и разработать алгоритмы, для решения задачи;
* разработать модульную структуру программы;
* спроектировать интерфейс;
* написать код программы;
* выполнить тестирование.

Объект разработки – программа.

Среда разработки – DosBox.

Методологической основой для написания программы послужило задание на курсовую работу.

**Основная часть**

**Изучение и описание предметной области**

По заданию на курсовую работу, необходимо написать программу на ассемблере, выполняющую вычисление значения заданной функции на указанном отрезке (с помощью сопроцессора)

Программа должна:

* вычислять значение заданной функции;
* выполнять вычисление на указанном отрезке;
* производить вычисление с помощью сопроцессора.

Одной из причин для разработки является отсутствие решений аналогичных задач.

Для выполнения задания нам понадобится DoxBox, а также, для компиляции, транслятор Turbo Assembler и компоновщик Turbo Linker.

**Постановка задачи**

Перечень исходных данных, их характеристики и способ представления:

1. значение x (ввод с клавиатуры, двойное слово).

Набор выполняемых функций:

1. расчёт значения степени двойки;
2. расчёт тангенса.

Перечень результатов, их характеристики и способ представления:

значение функции y (вывод на экран, слово).

Среда функционирования программного продукта:

1. Microsoft Windows Server 2003 или Microsoft Windows XP или Microsoft Windows 2000 с пакетом обновления 3 (SP3) или более поздние версии данных операционных систем;
2. Эмулятор DOSBox.

**Выбор структур данных для решения поставленной задачи**

Assembler — язык программирования низкого уровня, представляющий собой формат записи машинных команд, удобный для восприятия человеком.

Команды языка ассемблера один в один соответствуют командам процессора и, фактически, представляют собой удобную символьную форму записи (мнемокод) команд и их аргументов. Также язык ассемблера обеспечивает базовые программные абстракции: связывание частей программы и данных через метки с символьными именами и директивы.

Директивы ассемблера позволяют включать в программу блоки данных (описанные явно или считанные из файла); повторить определённый фрагмент указанное число раз; компилировать фрагмент по условию; задавать адрес исполнения фрагмента, менять значения меток в процессе компиляции; использовать макроопределения с параметрами и др.

Каждая модель процессора, в принципе, имеет свой набор команд и соответствующий ему язык (или диалект) ассемблера[1].

Таблица 1

Таблица идентификаторов

|  |  |  |
| --- | --- | --- |
| Наименование | Обозначение | Тип данных |
| Значение x | x | двойное слово |
| Константа 3 | a | двойное слово |
| Константа 4 | b | двойное слово |
| Константа 1 | c | двойное слово |
| Левый конец отрезка -1 | q1 | двойное слово |
| Правый конец отрезка -0.5 | q2 | двойное слово |
| Значение функции | y | слово |

**Логическое проектирование**

1. Определение и подключение сегмента данных, в котором объявляются переменные, используемые в программе.
2. Инициализация сопроцессора загрузка вещественного числа из памяти по адресу в него.
3. Расчёт значения для степени двойки.
4. Расчёт значения тангенса.
5. Вычисление значения функции.

Для написания программы, которая вычисляет значение функции на заданном промежутке, был разработан алгоритм, а по нему составлена Блок-схема. Блок-схема программы представлена на Рис. 1.

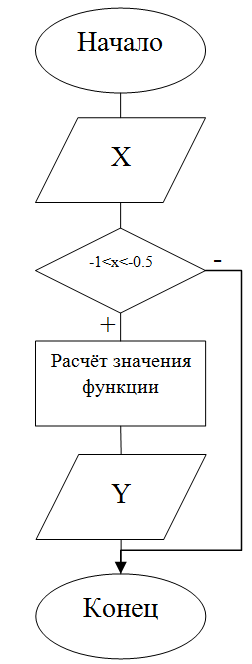


Рис. 1. Блок-схема

**Физическое проектирование программного обеспечения**

В процессе физического проектирования разработана модульная структура программы. Программа содержит два модуля и две функции. Модуль “DATA” отвечает за определение переменных. В модуле “CODE” выполняется проверка на принадлежность заданного значения к промежутку и вычисляется значение функции.

Таблица 2

Спецификация модуля DATA

|  |  |  |  |
| --- | --- | --- | --- |
| Имя модуля | Заголовок процедуры или функции | Формальные параметры | Выполняемое действие |
| DATA | DS:DATA | Значение x | Объявление значений переменных |

Таблица 3

Спецификация модуля CODE

|  |  |  |  |
| --- | --- | --- | --- |
| Имя модуля | Заголовок процедуры или функции | Формальные параметры | Выполняемое действие |
| CODE | CS:CODE | Значение x | Расчёт значения функции на заданном отрезке |

**Кодирование**

Реализация разработанных алгоритмов и составленный по ним текст программы (с комментариями) в *Приложении 2*.

**Тестирование**

На этапе тестирования разработаны тестовые данные и оформлены в виде таблицы.

Таблица 4

Таблица тестовых данных

|  |  |  |  |
| --- | --- | --- | --- |
| № | Исходные данные | Тестируемый модуль или подпрограммы | Ожидаемый результат |
| 1 | x=-0.7 | CODE | 0.30221700765 |
| 2 | x=-0.8 | CODE | -0.50042871025 |
| 3 | x=-0.9 | CODE | -1.40506620375 |
| 4 | x=-1.5 | CODE | Пропуск |
| 5 | x=-0.3 | CODE | Пропуск |

**Проведение тестирования по таблице тестовых данных**

Таблица 5

Результаты выполнения тестирования

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| № | Дата и время | Тестируемый модуль или подпрограмма | Кто проводил тестирование | Описание теста | Результаты тестирования |
| 1 | 15.12.2016 | CODE | Климов А.Г. | При x=-0.7 значение y=0.30211709933022333 | Успех |
| 2 | 15.12.2016 | CODE | Климов А.Г. | При x=-0.8 значение  y=-0.50042881086755245 | Успех |
| 3 | 15.12.2016 | CODE | Климов А.Г. | При x=-0.9 значение  y=-1.4050659707266385 | Успех |
| 4 | 15.12.2016 | CODE | Климов А.Г. | При x=-1.5 программа не выполняется | Успех |
| 5 | 15.12.2016 | CODE | Климов А.Г. | При x=-0.3 программа не выполняется | Успех |

**Заключение**

В последнее время широко используются различные программы для решения определённых задач, в том числе и на языках низкого уровня. Нахождение значения функции при помощи программных средств значительно упрощает и ускоряет вычисления.

При написании курсового проекта была изучена специальная литература, включающая в себя статьи, учебники, руководства, описаны теоретические аспекты и раскрыты ключевые понятия исследования, рассмотрено практическое применение программирования на ассемблере.

Во время написания курсовой работы был решён ряд поставленных задач:

* проанализировали задание на курсовую работу;
* определили требования к программному продукту;
* выбрали типы данных, необходимые для решения задачи;
* изучили и разработали алгоритмы, для решения задачи;
* разработали модульную структуру программы;
* спроектировали интерфейс;
* написали код программы;
* выполнили тестирование.

В том числе выполнено техническое задание.

Результатом курсовой работы стала программа, написанная на ассемблере.

**Источники информации**

1. Progopedia [Электронный ресурс] //Assembler. URL:

<http://progopedia.ru/language/assembler/> (дата обращения: 10.11.2016).

2. Программирование [Электронный ресурс] //Система команд сопроцессора. URL: <http://prog-cpp.ru/asm-coprocessor-command/> (дата обращения: 15.11.2016).

*ПРИЛОЖЕНИЕ 1*

МИНОБРНАУКИ РОССИИ

Федеральное государственное бюджетное образовательное учреждение высшего образования

«ЧЕРЕПОВЕЦКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»

наименование института (факультета)

наименование кафедры

наименование дисциплины в соответствии с учебным планом

УТВЕРЖДАЮ

Зав. кафедрой ,

д.т.н., профессор Ершов Е.В.

« » 20 г.

Программирование на языке низкого уровня

Техническое задание на курсовую работу

Листов

Руководитель

Ф.И.О. преподавателя

Исполнитель

студент

группа

Фамилия, имя, отчество

год

**Введение**

Программа на ассемблере, выполняющая вычисление значения заданной функции на указанном отрезке (с помощью сопроцессора).

**Основания для разработки**

Задание на Курсовую работу по Системному программированию. Череповецкий Государственный Университет 2016 год.

**Назначение разработки**

Программа должна вычислять значение функции  на отрезке . Результат работы должен отображаться в отладчике.

**Требования к программе**

*Требования к функциональным характеристикам:*

1. В сегменте данных должны описываться переменные, которые используются в программе.
2. Настройка на сегмент данных должна происходить следующим образом:

*MOV AX, DATA*

*MOV DS, AX*

1. Вычисление значения функции должно происходить с использованием сопроцессора, который инициализируется при помощи команды: finit.
2. Для вычисления значения выражения вида  должна использоваться следующая формула перевода: , которая вычисляется с использованием команд FYL2X, F2XM1, FSCALE и FPREM.
3. Запуск и компиляция программы должны происходить в эмуляторе DOSBox.
4. Для компиляции программы, написанной на языке ассемблера, должны использоваться транслятор Turbo Assembler (исполнительный файл TASM.EXE) и компоновщик Turbo Linker (исполнительный файл TLINK.EXE).

*Требования к надежности:*

* программа должна выполнять предписанные функциональные характеристики без сбоев;
* обеспечение контроля входной и выходной информации;
* защита при неверных действиях пользователя;
* контроль соответствия типов данных.

*Условия эксплуатации:*

* компьютер, поддерживающий операционную систему DOS или эмулятор DOSBox;
* наличие на компьютере операционной системы DOS или эмулятора DOSBox;
* знание основ работы в операционной системе DOS или эмуляторе DOSBox.

*Требования к составу и параметрам технических средств:*

* для установки DOSBox требуется Windows 98 или более поздняя версия и SVGA видеоадаптер;
* процессор 486DX/66 MHz или лучше;
* как минимум 16 Мб ОЗУ;
* не менее 195 Мб свободного дискового пространства;
* для установки DOSBox требуется устройство для чтения компакт-дисков (или же поддержка других устройств, таких как флэш-накопителей);
* необходим монитор Super VGA с разрешением 800x600 или более высоким, отображающий 256 и более цветов;
* необходима мышь Microsoft Mouse, Microsoft IntelliMouse или совместимое с ними указывающее устройство.

*Требования к информационной и программной совместимости:*

* язык программирования (Ассемблер);
* для компиляции, программы (транслятор Turbo Assembler и компоновщик Turbo Linker);
* операционная система (Windows);
* эмулятор операционной системы (DOSBox);
* уровень защиты (без защиты).

*Требования к маркировке и упаковке:*

Распространение на электронных носителях или через средства коммуникации (для некоммерческого использования).

*Требования к транспортированию и хранению:*

Без специальных требований.

**Требования к программной документации**

*Наличие различной документации:*

* руководства пользователя;
* текст программы;
* расчетно-пояснительная записка с приложениями.

**Технико-экономические показатели**

Преимущества разработки по сравнению с существующими отечественными и зарубежными аналогами – программа должна выполняется на языке низкого уровня и может использоваться на старом поколении компьютеров.

**Стадии и этапы разработки**

Таблица П1.1

Стадии и этапы разработки

|  |  |  |  |
| --- | --- | --- | --- |
| Наименование этапа разработки ПО | Сроки разработки | Результат выполнения | Отметка  о выполнении |
| Получение задания | 9.10.2016 | Выполнен успешно | Выполнено |
| Анализ требований | 10.10.16-12.10.16 | Выполнен успешно | Выполнено |
| Проектирование | 12.11.16-15.11.16 | Выполнен успешно | Выполнено |
| Реализация | 15.11.16-17.11.16 | Выполнен успешно | Выполнено |
| Тестирование | 13.12.16-17.12.16 | Выполнен успешно | Выполнено |
| Внедрение и поддержка | 18.12.16-30.12.16 | Выполнен успешно | Выполнено |

**Порядок контроля и приемки**

Таблица П1.2

Порядок контроля и приемки

|  |  |  |  |
| --- | --- | --- | --- |
| Наименование контрольного этапа выполнения курсовой работы | Сроки контроля | Результат выполнения | Отметка о приемке результата контрольного этапа |
| Техническое задание |  |  |  |
| Написание программы |  |  |  |
| Расчётно-пояснительная записка |  |  |  |
| Руководство пользователя |  |  |  |

*ПРИЛОЖЕНИЕ 2*

**Текст программы**

1. CODE SEGMENT ; сегмент команд
2. ASSUME CS:CODE, DS:DATA, SS:STAC
3. Start: MOV AX, DATA ; настройка DS
4. MOV DS, AX ; на сегмент данных
5. finit ; инициализация сопроцессора
6. fld x ; загрузка вещественного числа из памяти по адресу x в сопроцессор
7. fcomp q1 ; сравнение значений q1 и x
8. FSTSW SW ; загрузка регистра состояния в переменную SW
9. AND SW, 4500H ; наложение маски
10. CMP SW, 4000H ; проверка на установку флага C3 (=)
11. JE L4
12. JMP L1
13. L1:
14. fld x ; загрузка вещественного числа из памяти по адресу x в сопроцессор
15. fcomp q2 ; сравнение значений q2 и x
16. FSTSW SW ; загрузка регистра состояния в переменную SW
17. AND SW, 4500H ; наложение маски
18. CMP SW, 4000H ; проверка на установку флага C3 (=)
19. JE L4
20. JMP L2
21. L2:
22. fld x ; загрузка вещественного числа из памяти по адресу x в сопроцессор
23. fcomp q1 ; сравнение st0 и q1
24. FSTSW SW ; загрузка регистра состояния в переменную SW
25. AND SW, 4500H ; наложение маски
26. CMP SW, 0000H ; значение флагов C0, C2, C0 равно нулю (>)
27. JE L3
28. JMP L5
29. L3:
30. fld x ; загрузка вещественного числа из памяти по адресу x в сопроцессор
31. fcomp q2 ; сравнение st0 и q1
32. FSTSW SW ; загрузка регистра состояния в переменную SW
33. AND SW, 4500H ; наложение маски
34. CMP SW, 0100H ; установка флага C0 (<)
35. JE L4
36. JMP L5
37. L4:
38. fld x ; загрузка вещественного числа из памяти по адресу x в сопроцессор
39. fptan ; единица в st0 и значение тангенса в st1
40. fmul a ; умножение st0 на a
41. fmul ; умножение st0 на st1 (результат в st0)
42. fld x
43. F2XM1 ; 2^st0-1
44. fadd c ; 2^st0
45. fptan
46. fmul b ; умножение st0 на b
47. fmul
48. fadd st, st(1)
49. fist y
50. L5:
51. MOV AX, 4C00H ; функция завершения программы
52. INT 21H ; вызов DOS
53. CODE ENDS
54. DATA SEGMENT ; сегмент данных
55. x DD -0.9 ; определение переменной x
56. a DD 3.0 ; определение промежуточного числа
57. b DD 4.0 ; определение промежуточного числа
58. c DD 1.0 ; определение промежуточного числа
59. q1 DD -1.0 ; определение границы отрезка
60. q2 DD -0.5 ; определение границы отрезка
61. SW DW ? ; переменная для хранения регистра состояния
62. y DW ? ; значение функции
63. DATA ENDS
64. STAC SEGMENT PARA STACK ; сегмент стека
65. DB 128 DUP (?) ; область памяти под стек
66. STAC ENDS
67. END Start ; точка входа

*ПРИЛОЖЕНИЕ 3*

**Руководство пользователя**

**Общие сведения о программе**

Программа на ассемблере, выполняет вычисление значения заданной функции на указанном отрезке (с помощью сопроцессора):

 на отрезке .

**Описание установки**

Для использования программы потребуются:

1. Операционная система DOS или эмулятор DOSBox (для Windows);
2. транслятор Turbo Assembler (исполнительный файл TASM.EXE);
3. компоновщик Turbo Linker (исполнительный файл TLINK.EXE).

**Описание запуска программы в эмуляторе DOSBox**

Пример для программы, находящейся на диске d. Помимо файла программы 1.exe, должен присутствовать отладчик TD. EXE. Оба файла располагаются в одной папке.

Для загрузки программы в отладчик необходимо ввести ряд команд в эмуляторе DOSBox:

1. “mount d d:/” (монтирование образа диска);
2. “d:/” (переход в созданный образ);
3. “td 1.exe” (загрузка программы в отладчик).

После ввода каждой команды нажимается клавиша “Enter”. Пример правильно набранных команд на Рис П1.1.

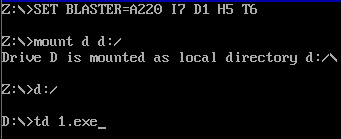


Рис. П3.1. Ввод команд

Если вы всё сделали верно, то откроется окно отладчика Рис П3.2.

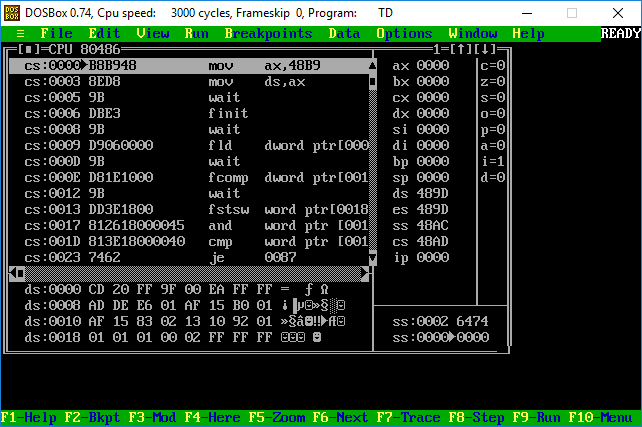


Рис. П3.2. Окно отладчика

Выполнение программы происходит при нажатии на клавишу F9. Вы можете проверить работу программы в пошаговом режиме. Для этого используется клавиша F7. Для возврата программы в исходное состояние необходимо нажать комбинацию клавиш Ctrl+F12.

Чтобы узнать результат выполнения, необходимо открыть вкладку сопроцессора. Для этого нажмите клавишу F10 и в разделе “View” выберите “Numeric processor”. Пример на Рис. П3.3.

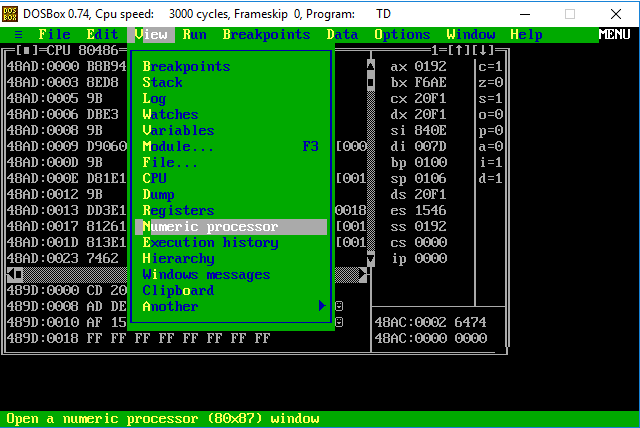


Рис. П3.3. Numeric processor

Затем, появится окно, в котором значение ST(0) будет являться искомым (значение y для заданной функции). В нашем случае, при x=-9 результат работы программы на Рис П3.4.

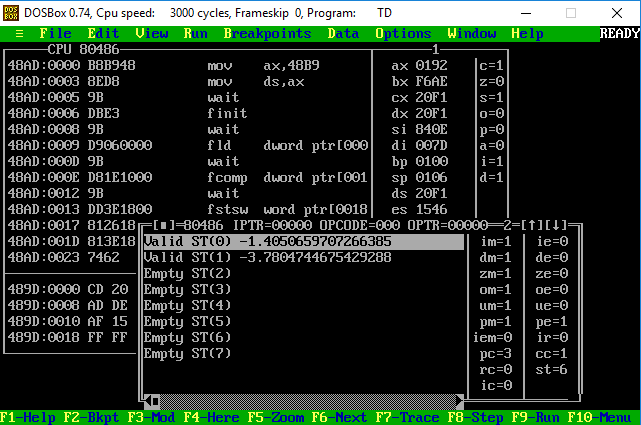


Рис. П3.4. Результат работы программы

**Инструкции по работе**

Пример для текста программы, находящегося на диске d. Изменение значения x происходит в файле текста программы 1.asm в сегменте данных. Пример на Рис. П3.5.



Рис. П3.5. Изменение значения x

После изменения значения x необходимо выполнять трансляцию и компоновку программы. Для этого в одной папке должны находиться файл с текстом программы 1.asm, транслятор TASM.EXE и компоновщик TLINK.EXE. Затем, последовательно выполняется ряд следующим команд в эмуляторе DOSBox:

1. “mount d d:/”;
2. “d:/”;
3. “tasm 1.asm”;
4. “tlink 1.obj”.

Пример правильного ввода команд на Рис П3.5.

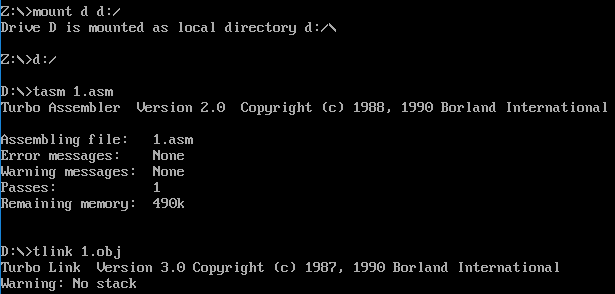


Рис. П3.5. Ввод команд

Результатом является файл программы 1.exe. О запуске программы вы можете узнать из прошлого пункта “Описание запуска программы в эмуляторе DOSBox”.