минобрнауки россии

федеральное государственное бюджетное

образовательное учреждение высшего образования

«ЧЕРЕПОВЕЦКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»

|  |  |
| --- | --- |
| Институт (факультет) | Институт Информационных Технологий |
| Кафедра | Математического и Программного Обеспечения ЭВМ |

**КУРСОВАЯ РАБОТА**

|  |
| --- |
| по дисциплине Программирование на ассемблере |

|  |  |
| --- | --- |
| на тему | Программирование на языке низкого уровня |

|  |
| --- |
| Выполнил студент группы |
| 1ПИб-02-3оп-22 |
| направление подготовки (специальности) |
| 09.03.04, Программная инженерия |
| *шифр, наименование* |
| Маркелов Сергей Александрович |
| *фамилия, имя, отчество* |

|  |
| --- |
| Руководитель |
| Виноградова Людмила Николаевна |
| *фамилия, имя, отчество* |
| доцент |
| *должность* |

|  |
| --- |
| Дата представления работы |
| «\_\_21\_\_»\_\_\_\_\_\_\_декабря\_\_\_\_\_2023 г. |
|  |
| Заключение о допуске к защите |
|  |
|  |
|  |
|  |
| Оценка \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_, \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ |
| количество баллов |
| Подпись преподавателя\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ |

Череповец, 2023

*Год*

Аннотация

Данную курсовую работу по дисциплине «Программирование на ассемблере» на тему «Программирование на языке низкого уровня» выполнил студент группы 1ПИб-02-3оп-22 Института информационных технологий Череповецкого государственного университета Маркелов Сергей Александрович.

Целью курсовой работы является знакомство с принципами работы с языками низкого уровня, а именно с языком программирования Assembler для микропроцессора Intel 8086.

Курсовая работа содержит описание разработки программы, обрабатывающей массив типа структуры, в котором содержится информация о 10 кулинарных блюдах. Программа подсчитывает количество блюд без мяса и записывает это число в регистр DI.

Оглавление

[Введение 4](#_Toc154786891)

[1. Изучение и описание предметной области 6](#_Toc154786892)

[2. Постановка задачи 8](#_Toc154786893)

[3. Выбор структур данных 9](#_Toc154786894)

[4. Логическое проектирование 10](#_Toc154786895)

[5. Физическое проектирование 13](#_Toc154786896)

[6. Кодирование 14](#_Toc154786897)

[7. Тестирование 16](#_Toc154786898)

[Заключение 19](#_Toc154786899)

[Список литературы 20](#_Toc154786900)

[Приложение 1. Техническое задание 21](#_Toc154786901)

[Приложение 2. Руководство пользователя 29](#_Toc154786902)

[Приложение 3. Текст программы 35](#_Toc154786903)

# Введение

Программы играют огромную роль в современном мире. Существует большое количество различных видов программ: драйвера, утилиты, плееры, браузеры, различные редакторы (текстовые, графические, музыкальные видеоредакторы), мессенджеры, архиваторы, антивирусы и многие другие.

Благодаря программам мы можем смотреть, обрабатывать или создавать фотографии, видео, музыку, работать с документами, обрабатывать данные и многое другое.

Программы создаются благодаря языкам программирования – формальным знаковым системам, предназначенным для описания алгоритмов обработки данных. Каждый язык программирования имеет свой алфавит и синтаксис [[3]](#_Источники).

Языки программирования разделяются на языки высокого уровня и языки низкого уровня. Уровень языка обозначает его удаленность от машинного кода целевой архитектуры процессора. Низкий уровень означает меньший масштаб преобразований, которые должен претерпеть код программы перед тем, как он может быть запущен. В данной курсовой работе будет рассматриваться язык программирования низкого уровня [3].

Одним из наиболее популярных языков программирования низкого уровня является Assembler (Ассемблер). Ассемблер представляет собой промежуточное звено между машинным кодом и языками программирования высокого уровня. Этот язык используется для написания программного обеспечения, которое управляет компьютером на более низком уровне, непосредственно взаимодействуя с аппаратным обеспечением [4].

Программы на языке ассемблера представляют собой набор инструкций, каждая из которых соответствует определенной команде процессора. Затем эти инструкции транслируются, или, по-другому, ассемблируются в машинный код – набор двоичных чисел, которые распознает центральный процессор и выполняет соответствующие введенным инструкциям операции.

При этом, для каждого процессора существует свой собственный язык ассемблера. В данной курсовой работе будет рассматриваться язык ассемблера для процессоров, созданных на базе архитектуры микропроцессора Intel 8086, выпущенного компанией Intel в 1978 году. В процессорах данной фирмы реализована преемственность, поэтому программы, написанные для младшей модели, могут быть без изменений запущены на более старшей.

Целью курсовой работы является знакомство с принципами работы с языками низкого уровня, а именно с языком программирования Assembler для микропроцессора Intel 8086.

В рамках курсовой работы необходимо написать программу на языке Ассемблер, которая обрабатывает массив типа структуры с информацией о блюдах и подсчитывает количество блюд без мяса.

Для создания программы использовались транслятор Turbo Assembler, компоновщик Turbo Linker и эмулятор операционной системы MS-DOS DOSBox.

# Изучение и описание предметной области

Для реализации разрабатываемой программы необходимо использовать массивы и структуры данных.

Массив – это структура данных, которая хранит набор значений какого-либо одного типа (элементов массива), идентифицируемых по одному или нескольким индексам. В зависимости от количества индексов определяется размерность массива: одномерные – с одним индексом, двумерные – с двумя и т. д. В разрабатываемой программе необходимо использовать одномерный массив [2].

Массивы в ассемблере описываются по директивам определения данных. Допускается использование конструкции повторения DUP:

A DB 1, 2, 3 ; определение массива A типа BYTE с начальными значениями 1, 2, 3

B DW 10 DUP (5) ; определение массива B типа WORD из 10 элементов с повторяющимся начальным значением 5

Чтобы обратиться к элементу массива, нужно указать адрес начала массива и смещение элемента в массиве, например, A[2] или B[BX]. Смещение первого элемента массива равно 0. Смещения остальных элементов массива зависят от размера элементов.

Структура – это составной тип данных, занимающий несколько соседних ячеек памяти. Структура состоит из нескольких различных компонентов, называемых полями структуры. При этом, в отличие от элементов массива, поля могут быть разного типа (размера) [2].

Тип структуры описывается следующим образом:

TIME STRUC ; TIME – имя типа, STRUC – директива для описания структуры

HOUR DB 12 ; определение поля HOUR типа BYTE с начальным значением 12

MIN DB 40 ; определение поля MIN типа BYTE с начальным значением 40

SEC DB 37 ; определение поля SEC типа BYTE с начальным значением 37

TIME ENDS ; окончание описания типа структуры

После описания типа структуры можно описывать переменные данного типа. Эти переменные называются структурами. Описываются они следующим образом:

T TIME <20, 39, 47> ; определение переменной T типа структуры TIME со значениями по умолчанию <20, 39, 47>

Чтобы обратиться к полю структуры, нужно указать название переменной и название поля, например, T.HOUR.

В ассемблере удобнее одной директивой описывать сразу несколько структур с помощью массива, элементами которого являются структуры. Для этого в директиве указывается несколько операндов и/или конструкция повторения DUP, например:

T TIME <13, 27, 41>, <15, 18, 12>, <17, 30, 25> ; определение массива T, элементами которого являются переменные типа структуры TIME со значениями по умолчанию <13, 27, 41>, <15, 18, 12>, <17, 30, 25>

T TIME <19, 25, 12>, 5 DUP (<>) ; определение массива T, элементами которого являются переменные типа структуры TIME, первая из которых имеет значение по умолчанию <19, 25, 12>, а остальные 5 не определены

# Постановка задачи

Для реализации программы необходимо написать код на языке ассемблера. Он должен содержать структуру FOOD, содержащую информацию о блюдах и состоящую из полей:

* DISH – название блюда (строка, тип BYTE);
* PRODUCT – главный ингредиент блюда (строка, тип BYTE);
* MEAT – наличие мяса (число типа BYTE, может принимать значения 0 – нет мяса, и 1 – есть мясо);
* CCAL – калорийность блюда (число типа WORD).

Программа должна содержать информацию о 10 блюдах. Их необходимо записать в массив D типа структуры FOOD.

Программа перебирает элементы массива и подсчитывает количество блюд без мяса, т. е. количество структур, в которых поле MEAT имеет значение 0. Полученный результат должен быть записан в регистр DI.

# Выбор структур данных

Структуры данных, которые были использованы в программе, представлены в табл. 1.

Таблица 1

Структуры данных

|  |  |  |
| --- | --- | --- |
| Наименование | Обозначение | Тип данных |
| Структура | FOOD | STRUC |
| Поле структуры | DISH | BYTE |
| Поле структуры | PRODUCT | BYTE |
| Поле структуры | MEAT | BYTE |
| Поле структуры | CCAL | WORD |
| Массив | D | FOOD |

# Логическое проектирование

Разрабатываемая программа выполняет следующий алгоритм:

1. В сегменте DATA определяется структура FOOD и массив D;
2. Сегмент DATA записывается в регистр AX;
3. Значение регистра AX записывается в регистр DS;
4. Значения регистров AX, BX и DI обнуляются с помощью директивы XOR (исключающее ИЛИ);
5. В регистр CX записывается значение 10. В дальнейшем этот регистр будет играть роль счетчика в цикле;
6. Значение значения регистра CX сравнивается с нулем с помощью команды сравнения CMP;
7. Если значение регистра CX больше нуля, с помощью оператора перехода JA (больше) процессор передает управление в метку M1. Иначе – безусловный переход (с помощью директивы JMP) в метку stop.
8. Метка M1:
   1. Обнуляется значение регистра DL с помощью директивы XOR (исключающее ИЛИ);
   2. В регистр DL записывается значение поля MEAT структуры, которая является элементом массива D с номером – значением регистра;
   3. Значение регистра DL сравнивается с нулем с помощью команды сравнения CMP;
   4. Если значение регистра DL равно нулю, с помощью оператора перехода JE (равно) процессор передает управление в метку M2. Иначе – продолжается выполнение метки M1;
   5. Значение регистра BX увеличивается на число, равное размеру структуры (команда TYPE FOOD) с помощью директивы сложения ADD;
   6. Значение регистра CX (счетчик цикла) уменьшается на 1 с помощью декремента (директива DEC);
   7. Значение значения регистра CX сравнивается с нулем с помощью команды сравнения CMP;
   8. Если значение регистра CX больше нуля, с помощью оператора перехода JA (больше) процессор передает управление в метку M1. Иначе – безусловный переход (с помощью директивы JMP) в метку stop.
9. Метка M2:
   1. Значение регистра DI увеличивается на 1 с помощью инкремента (директива INC);
   2. Значение регистра BX увеличивается на число, равное размеру структуры (команда TYPE FOOD) с помощью директивы сложения ADD;
   3. Значение регистра CX (счетчик цикла) уменьшается на 1 с помощью декремента (директива DEC);
   4. Значение значения регистра CX сравнивается с нулем с помощью команды сравнения CMP;
   5. Если значение регистра CX больше нуля, с помощью оператора перехода JA (больше) процессор передает управление в метку M1. Иначе – безусловный переход (с помощью директивы JMP) в метку stop.
10. Метка stop: вызывается прерывание INT 21h.

Блок-схема данной программы представлена на рис. 1.



Рис. 1. Блок-схема программы

# Физическое проектирование

Программа состоит из одного файла KP.asm, в котором содержится код программы на языке ассемблера. Процедуры и функции в программе не используются.

Спецификация меток, использованных в программе, отображена в табл. 2.

Таблица 2

Спецификация меток

|  |  |  |  |
| --- | --- | --- | --- |
| Имя | Тип | Формальные параметры | Назначение |
| Start | - | - | Установка начальных значений регистров |
| M1 | - | - | Обработка массива, запись значений элементов в регистр DL |
| M2 | - | - | Запись количества блюд без мяса в регистр DI |
| stop | - | - | Вызов прерывания INT 21h, окончание программы |

# Кодирование

В начале программы описывается структура данных FOOD:

FOOD STRUC

DISH DB 255 DUP (?)

PRODUCT DB 255 DUP (?)

MEAT DB ?

CCAL DW ?

FOOD ENDS

Далее описывается массив D типа структуры FOOD, ему задаются значения по умолчанию:

D FOOD<'Seld pod shuboy','Seld',0,189>, <'Olivye','Kolbasa',1,150>, <'Kartofel pecheniy', 'Kartofel', 0, 145>, …

Затем в сегменте CODE регистрам присваиваются начальные значения.

MOV AX, @DATA

MOV DS, AX

XOR AX, AX

MOV CX, 10

XOR BX, BX

XOR DI, DI

Значение регистра CX (счётчик цикла) сравнивается с нулем. Если он больше нуля – управление передается в метку М1, в противном случае – управление передается в метку stop:

CMP CX, 0

JA M1

JMP stop

Описывается метка M1 – цикл, в котором определяется, есть ли мясо в блюде:

M1:

XOR DL, DL

MOV DL, D.MEAT[BX]

Значение регистра DL сравнивается с нулем. Если он равен нулю – управление передается в метку М2. Иначе – значение регистра CX (счетчик цикла) сравнивается с нулем. Если он больше нуля – управление передается обратно в начало метки M1, в противном случае – управление передается в метку stop:

CMP DL, 0

JE M2

ADD BX, TYPE FOOD

DEC CX

CMP CX, 0

JA M1

JMP stop

Описывается метка M2 – цикл, в котором в регистр DI записывается количество блюд без мяса:

M2:

INC DI

ADD BX, TYPE FOOD

Значение регистра CX (счетчик цикла) сравнивается с нулем. Если он больше нуля – управление передается обратно в метку M1, в противном случае – управление передается в метку stop:

DEC CX

CMP CX, 0

JA M1

JMP stop

В метке stop вызывается прерывание. Программа завершается:

stop:

MOV AX, 4C00h

INT 21h

Полный код программы представлен в приложении 3.

# Тестирование

Наборы тестовых данных представлены в табл. 3.

Таблица 3

Тестовые данные

|  |  |  |  |
| --- | --- | --- | --- |
| № | Исходные данные | Тестируемый модуль или подпрограмма | Ожидаемый результат |
| 1 | 2 | 3 | 4 |
| 1 | <'Seld pod shuboy','Seld',0,189>, <'Olivye','Kolbasa',1,150>, <'Kartofel pecheniy', 'Kartofel', 0, 145>, <'Myaso po-francuzski', 'Myaso', 1, 264>, <'Kraboviy salat', 'Krabovie palochki', 0, 128>, <'Buterbrod s ikroy', 'Ikra', 0, 180>, <'Holodec', 'Myaso', 1, 350>, <'Zalivnaya riba', 'Riba', 0, 100>, <'Salat Mimoza', 'Kartofel', 0, 183>, <'Shokoladniy pirog', 'Muka', 0, 478> | kp.asm | 7 |
| 2 | <'Cezar','Salat',1,218>, <'Ovoshnoy salat', 'Ogurets', 0, 20>, <'Yaichnitsa s kolbasoy','Yaico',1,230>, <'Sharlotka', 'Muka', 0, 145>, <'Kuritsa pechenaya', 'Kuritsa', 1, 223>, <'Lazanya', 'Makaroni', 1, 244>, <'Golubci', 'Kapusta', 1, 139>, <'Cheburek', 'Muka', 1, 279>, <'Mannaya kasha', 'Mannaya krupa', 0, 83>, <'Shashlik', 'Myaso', 1, 180> | kp.asm | 3 |
| 3 | <'Cezar','Salat',1,218>, <'Yaichnitsa s kolbasoy','Yaico',1, 230>, <'Lazanya', 'Makaroni', 1, 244>, <'Golubci', 'Kapusta', 1, 139>, <'Cheburek', 'Muka', 1, 279>, <'Shashlik', 'Myaso', 1, 180>, <'Olivye','Kolbasa',1,150>, <'Holodec', 'Myaso', 1, 350>, <'Myaso po-francuzski', 'Myaso', 1, 264>, <'Kuritsa pechenaya', 'Kuritsa', 1, 223> | kp.asm | 0 |

Продолжение табл. 3

|  |  |  |  |
| --- | --- | --- | --- |
| 1 | 2 | 3 | 4 |
| 4 | <'Seld pod shuboy','Seld',0,189>, <'Kartofel pecheniy', 'Kartofel', 0, 145>, <'Kraboviy salat', 'Krabovie palochki', 0, 128>, <'Buterbrod s ikroy', 'Ikra', 0, 180>, <'Zalivnaya riba', 'Riba', 0, 100>, <'Salat Mimoza', 'Kartofel', 0, 183>, <'Shokoladniy pirog', 'Muka', 0, 478>, <'Ovoshnoy salat', 'Ogurets', 0, 20>, <'Mannaya kasha', 'Mannaya krupa', 0, 83>, <'Sharlotka', 'Muka', 0, 145> | kp.asm | 10 |
| 5 | <'Seld pod shuboy','Seld',0,189>, <'Kartofel pecheniy', 'Kartofel', 0, 145>, <'Kraboviy salat', 'Krabovie palochki', 0, 128>, <'Holodec', 'Myaso', 1, 350>, <'Salat Mimoza', 'Kartofel', 0, 183>, <'Cezar','Salat',1,218>, <'Yaichnitsa s kolbasoy','Yaico',1,230>, <'Kuritsa pechenaya', 'Kuritsa', 1, 223>, <'Golubci', 'Kapusta', 1, 139>, <'Mannaya kasha', 'Mannaya krupa', 0, 83> | kp.asm | 5 |

Результаты тестов представлены в таблице 4.

Таблица 4

Результаты тестирования

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Дата тестирования | Тестируемый модуль | Тестирование проводил | Описание теста | Результат тестирования |
| 1 | 2 | 3 | 4 | 5 |
| 13.12.2023 | kp.asm | Маркелов С. А. | Тестирование с набором данных №1. | Ошибка. В регистр DI записано неверное значение |
| 14.12.2023 | kp.asm | Маркелов С. А. | Тестирование с набором данных №1 | Успех |

Продолжение табл. 4

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 1 | 2 | 3 | 4 | 5 |
| 15.12.2023 | kp.asm | Маркелов С. А. | Тестирование с набором данных №2. В регистр DI записано неверное значение | Ошибка. В регистр DI записано неверное значение |
| 16.12.2023 | kp.asm | Маркелов С. А. | Тестирование с набором данных №2 | Успех |
| 16.12.2023 | kp.asm | Маркелов С. А. | Повторное тестирование с набором данных №1 | Успех |
| 17.12.2023 | kp.asm | Маркелов С. А. | Тестирование с набором данных №3 | Успех |
| 18.12.2023 | kp.asm | Маркелов С. А. | Тестирование с набором данных №4 | Успех |
| 19.12.2023 | kp.asm | Маркелов С. А. | Тестирование с набором данных №5 | Успех |

# Заключение

В ходе выполнения курсовой работы была разработана программа, которая работает с массивом типа структуры, в котором записана информация о 10 блюдах. Программа подсчитывает количество блюд без мяса и записывает это число в регистр DI.

В ходе разработки программы были освоены базовые возможности языка Assembler для процессоров на базе архитектуры микропроцессора Intel 8086, а также приобретены навыки работы с транслятором Turbo Assembler и компоновщиком Turbo Linker.

# Список литературы

1. Методика и организация самостоятельной работы студентов: учебно-методическое пособие / Е.В. Ершов, Л.Н. Виноградова, В.В. Селивановских [и др.]. – Череповец: ФГБОУ ВПО ЧГУ, 2015. – 243 с.
2. Виноградова, Л. Н. Системное программирование: Учеб. Пособие. – Череповец: ФГБОУ ВПО ЧГУ, 2016. – 210 с.
3. Язык программирования — Википедия [электр.ресурс] <https://ru.wikipedia.org/wiki/Язык_программирования>. Дата обращения: 20.12.2023.
4. Ассемблер — SkillFactory [электр.ресурс] https://blog.skillfactory.ru/glossary/assembler. Дата обращения: 20.12.2023.

# Приложение 1

МИНОБРАНАУКИ РОССИИ

федеральное государственное бюджетное

образовательное учреждение высшего образования

ЧЕРЕПОВЕЦКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

Институт информационных технологий

наименование института (факультета)

Математическое и программное обеспечение ЭВМ

наименование кафедры

Программирование на ассемблере

наименование дисциплины в соответствии с учебным планом

УТВЕРЖДАЮ

Зав. кафедрой МПО ЭВМ

д. т.н. \_\_\_\_\_\_\_\_\_ Ершов Е.В.

«\_\_\_\_» \_\_\_\_\_\_\_\_\_\_\_2023 г.

Программирование на языке низкого уровня

Техническое задание на курсовую работу

Листов 8

Руководитель: Виноградова Л. Н.

Исполнитель: студент гр. 1ПИб-02-3оп-22

Маркелов С. А.

2023 г.

Введение

Курсовая работа направлена на разработку программы на языке программирования низкого уровня.

1. Основания для разработки

Основанием для разработки является задание на курсовую работу по дисциплине «Программирование на ассемблере», выданное на кафедре МПО ЭВМ ИИТ ЧГУ.

Дата утверждения: 9 ноября 2023 года.

Наименование темы разработки: «Программирование на языке низкого уровня».

2. Назначение разработки

Основной задачей курсовой работы является создание программы, которая обрабатывает массив D типа структуры данных Food с информацией о кулинарных блюдах, подсчитывает количество блюд без мяса и записывает его в регистр DI.

3. Требования к программе

* 1. Требования к функциональным характеристикам

К разрабатываемой программе предъявляются следующие требования:

* программа должна обрабатывать массив D типа структуры данных FOOD с использованием функций DOS;
* структура FOOD должна содержать поля «Наименование блюда», «Главный ингредиент», «Наличие мяса», «Калорийность»;
* поля «Наименование блюда» и «Главный ингредиент» должны представлять собой строки типа BYTE;
* поле «Наличие мяса» должно представлять собой беззнаковое число типа BYTE и принимать значения 0 или 1;
* поле «Калорийность» должно представлять собой беззнаковое число типа WORD;
* описание типа структуры может размещаться в любом месте программы, но обязательно до описания переменных этого типа;
* имена полей не должны совпадать с именами других объектов программы;
* не допускается вложенность структур;
* в массив D должна быть записана информация о 10 блюдах;
* программа должна записывать в регистр DI число – количество блюд без мяса в данном массиве.
  1. Требования к надежности

Программа должна работать без ошибок. Для этого после создания она должна быть протестирована. При возникновении каких-либо ошибок они должны быть исправлены.

* 1. Условия эксплуатации

Для корректной работы программы необходимо:

* наличие любого устройства, на котором возможно запускать файлы формата .asm (компьютер, ноутбук);
* наличие на устройстве современной версии любой операционной системы.
  1. Требования к составу и параметрам технических средств

Для корректной работы программы необходимо:

* оперативная память: не менее 1 Гб для 32-разрядной ОС, 2 Гб для 64-разрядной ОС;
* процессор на базе архитектуры микропроцессора 8086 не менее чем с 2 ядрами и тактовой частотой не ниже 1 ГГц;
* разрешение экрана не менее 800 х 600;
* видеокарта с видеопамятью не менее 1 Гб;
* наличие мыши и клавиатуры.
  1. Требования к информационной и программной совместимости

Для корректной работы программы необходимо:

* установленная операционная система (например, Windows XP, Vista, 7, 8, 8.1, 10, 11, macOS, Linux);
* установленный на устройстве транслятор «Turbo Assembler»;
* установленный на устройстве компоновщик «Turbo Linker»;
* установленный на устройстве эмулятор операционной системы MS-DOS «DOSBox».
  1. Требования к маркировке и упаковке

Требования к маркировке и упаковке программы не предъявляются.

* 1. Требования к транспортированию и хранению

Программа может храниться и транспортироваться в виде файлов, необходимых для ее работы, на носителе информации (флешка, CD-диск). Такую программу можно распространять, передавая его с одного компьютера на другой с помощью флешки или диска, либо отправляя ее по электронной почте.

* 1. Специальные требования

Специальные требования к программе не предъявляются.

4. Требования к программной документации

* 1. Содержание расчетно-пояснительной записки

Программная документация должна содержать расчётно-пояснительную

записку с содержанием:

1. Титульный лист
2. Аннотация
3. Оглавление
4. Введение
5. Основная часть
   1. Изучение и описание предметной области
   2. Постановка задачи
   3. Выбор структур данных
   4. Логическое проектирование
   5. Физическое проектирование
   6. Кодирование
   7. Тестирование
6. Заключение
7. Список литературы
8. Приложения
   1. Техническое задание
   2. Руководство пользователя
   3. Текст программы
   4. Требования к оформлению

В данном пункте представлены требования к оформлению документации в соответствии с ГОСТ (табл. П1.1).

Таблица П1.1

Требования к оформлению

|  |  |
| --- | --- |
| Документ | Печать на отдельных листах формата А4 (210х297 мм); оборотная сторона не заполняется; листы нумеруются. Печать возможна ч/б. Файлы предъявляются на компакт-диске: РПЗ с ТЗ; программный код. Листы и диск в конверте вложены в пластиковую папку скоросшивателя. |
| Страницы | Ориентация – книжная; отдельные страницы, при необходимости, альбомная. Поля: верхнее, нижнее – по 2 см, левое – 3 см, правое – 1 см. |
| Абзацы | Межстрочный интервал – 1,5, перед и после абзаца – 0. |
| Шрифты | Кегль – 14. В таблицах шрифт 12. Шрифт кода программы – 10 (возможно в 2 колонки). |
| Рисунки | Подписывается под ним по центру: Рис.Х. Название. В приложениях: Рис.П1.3. Название. |
| Таблицы | Подписывается: над таблицей, выравнивание по правому: «Таблица Х». В следующей строке по центру Название Надписи в «шапке» (имена столбцов, полей) – по центру. В теле таблицы (записи) текстовые значения – выровнены по левому краю, числа, даты – по правому. |

5. Технико-экономические показатели

Требования не предъявляются.

6. Стадии и этапы разработки

В данном пункте описаны стадии и этапы разработки программы (табл. П1.2).

Таблица П1.2

Стадии и этапы разработки

|  |  |  |  |
| --- | --- | --- | --- |
| Наименование этапа разработки | Сроки разработки | Результат выполнения | Отметка о выполнении |
| Определение темы для курсовой работы | 09.11.2023 | Утверждена тема для разработки |  |
| Оформление технического задания | 17.11.2023 | Оформленное техническое задание |  |
| Изучение и описание предметной области | 21.11.2023 | Получены теоретические знания для разработки программы |  |
| Выбор структур данных | 28.11.2023 | Выбраны структуры данных |  |
| Логическое проектирование | 05.12.2023 | Подготовлен проект логической части программы |  |
| Физическое проектирование | 12.12.2023 | Подготовлен проект физической части программы |  |
| Написание программы | 18.12.2023 | Готовая программа |  |
| Тестирование и отладка | 19.12.2023 | Конечный вариант программы |  |
| Оформление сопроводительной документации | 20.12.2023 | Оформленная сопроводительная документация |  |

7. Порядок контроля и приемки

В данном пункте описан порядок контроля и приемки курсовой работы (табл. П1.3).

Таблица П1.3

Порядок контроля и приемки

|  |  |  |  |
| --- | --- | --- | --- |
| Наименование контрольного этапа выполнения курсовой работы | Сроки контроля | Результат выполнения | Отметка о приемке результата контрольного этапа |
| Сдача технического задания | 17.11.2023 | Согласованное техническое задания |  |
| Сдача расчетно-пояснительной записки | 21.12.2023 | Согласованная расчетно-пояснительная записка |  |
| Сдача курсовой работы | 21.12.2023 | Получение оценки за выполненную работу |  |

# Приложение 2

Руководство пользователя

1. Общие сведения о программе

Файл программы называется «KP.exe». При запуске программа перебирает массив с информацией о кулинарных блюдах и записывает в регистр DI количество блюд без мяса.

1. Описание установки

Установка программы не требуется. Однако для ее запуска потребуется установить эмуляторе операционной системы MS-DOS.

1. Описание запуска

Программу необходимо запускать в эмуляторе операционной системы MS-DOS. Например, в программе «DOSBox».

Для начала необходимо смонтировать виртуальный диск с помощью команды mount и привязать к нему директорию с файлом кода на диске компьютера, например, «mount d: c:\asm», где d – название виртуального диска, c:\asm – расположение программы.

Затем нужно открыть смонтированный виртуальный диск, введя его название, например, «d:» (рис. П2.1).

После этих действий можно запустить программу в отладчике с помощью команды «td kp», где kp – название программы (рис. П2.2).

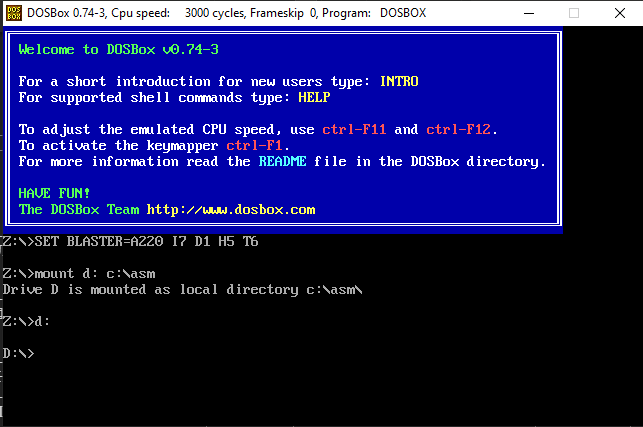


Рис. П2.1. Монтирование виртуального диска

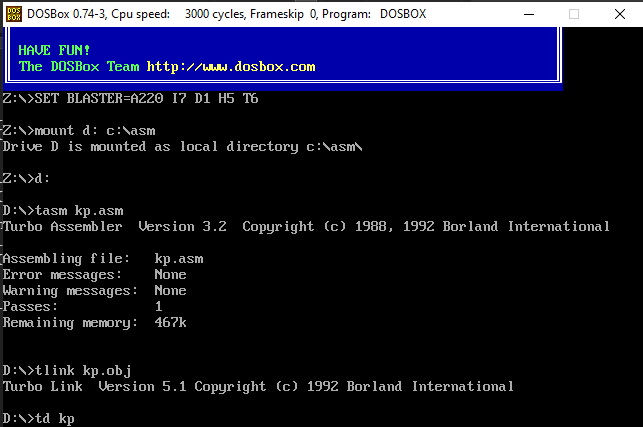


Рис. П2.2. Трансляция и компоновка программы, запуск в отладчике

1. Инструкции по работе

После запуска отладчика можно переходить к работе с программой (рис. П2.3). Программу можно выполнять пошагово, нажимая на клавишу F7 до тех пор, пока в коде не встретится прерывание. Либо можно выполнить всю программу сразу, нажав на клавишу F9. После выполнения программы количество блюд без мяса будет записано в регистр DI (рис П2.4).

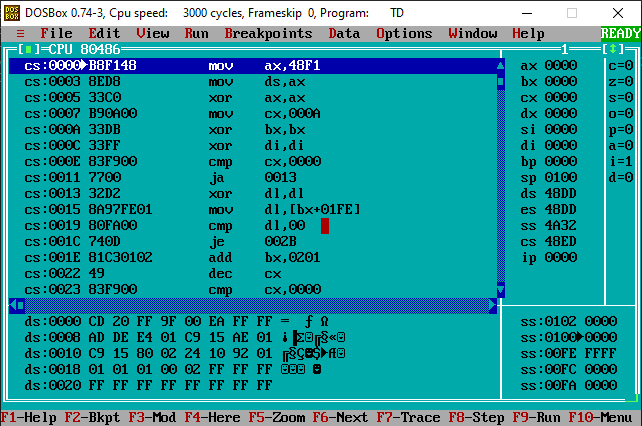


Рис. П2.3. Отладчик DOSBox в начале работы программы

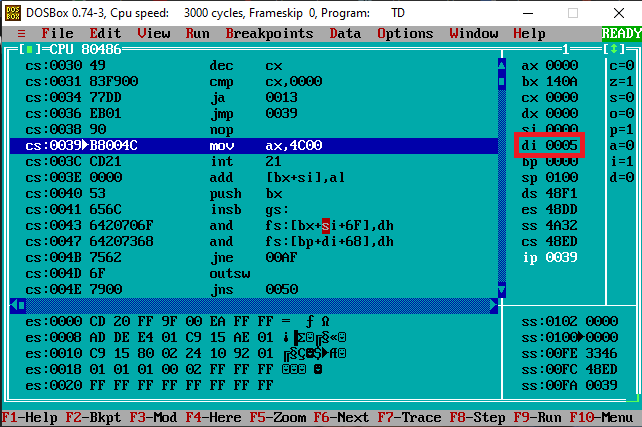


Рис. П2.4. Результат работы программы в регистре DI

1. Сообщения пользователю

При указании несуществующей директории файла появится сообщение «Directory … doesn’t exist!» («Директория … не существует!») (рис. П2.5). В случае появления такого сообщения нужно перепроверить правильность указания директории, а также её наличие на диске, и ввести ее еще раз.

При попытке открыть еще не смонтированный диск появится сообщение «Drive … does not exist!» («Диск … не существует!») (рис. П2.6). В случае появления такого сообщения нужно перемонтировать диск и попробовать открыть его еще раз.

При попытке запустить отладчик программы до привязки диска будет выведено сообщение «Illegal command: td» («Недопустимая команда: td») (рис. П2.7). В таком случае необходимо смонтировать диск и привязать к нему директорию с файлом программы.

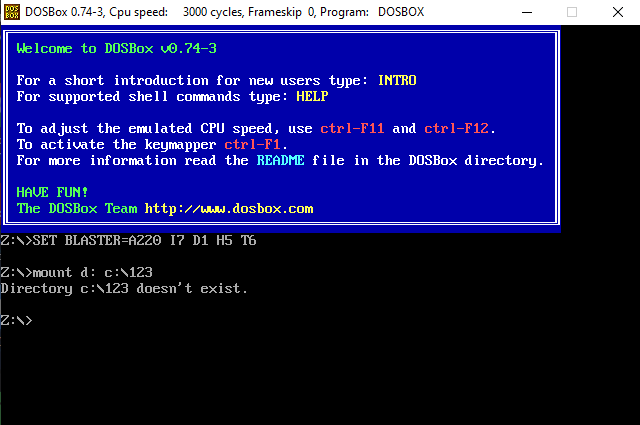


Рис. П2.5. Сообщение «Directory … doesn’t exist»

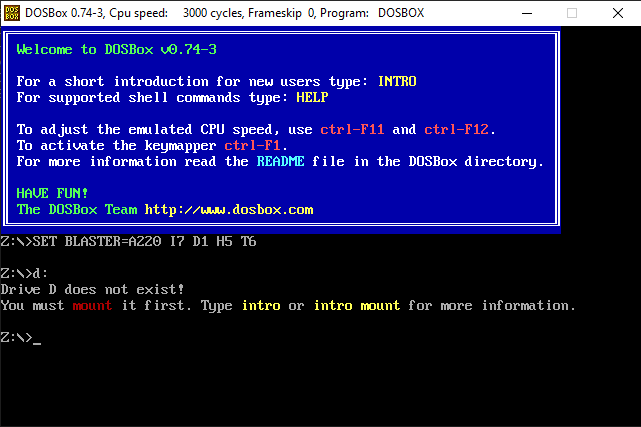


Рис. П2.6. Сообщение «Drive … does not exist»

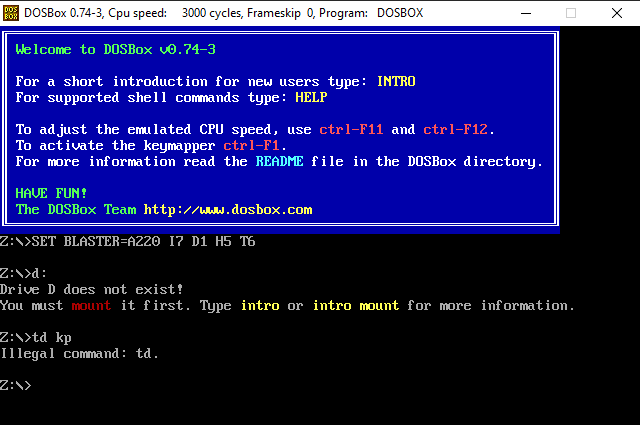


Рис П2.7. Сообщение «Illegal command: td»

# Приложение 3

Текст программы

.MODEL LARGE

.STACK 100H

.DATA

FOOD STRUC

DISH DB 255 DUP (?)

PRODUCT DB 255 DUP (?)

MEAT DB ?

CCAL DW ?

FOOD ENDS

D FOOD<'Seld pod shuboy','Seld',0,189>, <'Olivye','Kolbasa',1,150>, <'Kartofel pecheniy', 'Kartofel', 0, 145>, <'Myaso po-francuzski', 'Myaso', 1, 264>, <'Kraboviy salat', 'Krabovie palochki', 0, 128>, <'Buterbrod s ikroy', 'Ikra', 0, 180>, <'Holodec', 'Myaso', 1, 350>, <'Zalivnaya riba', 'Riba', 0, 100>, <'Salat Mimoza', 'Kartofel', 0, 183>, <'Shokoladniy pirog', 'Muka', 0, 478>

.CODE

Start:

MOV AX, @DATA

MOV DS, AX

XOR AX, AX

MOV CX, 10

XOR BX, BX

XOR DI, DI

CMP CX, 0

JA M1

JMP stop

M1:

XOR DL, DL

MOV DL, D.MEAT[BX]

CMP DL, 0

JE M2

ADD BX, TYPE FOOD

DEC CX

CMP CX, 0

JA M1

JMP stop

M2:

INC DI

ADD BX, TYPE FOOD

DEC CX

CMP CX, 0

JA M1

JMP stop

stop:

MOV AX, 4C00h

INT 21h

END Start